Synthesis and Analysis of Design Methods in Linear Repetitive, Iterative Learning and Model Predictive Control

Jianzhong Zhu

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2018

ABSTRACT

Synthesis and Analysis of Design Methods in Linear Repetitive, Iterative Learning and Model Predictive Control

Jianzhong Zhu

Repetitive Control (RC) seeks to converge to zero tracking error of a feedback control system performing periodic command as time progresses, or to cancel the influence of a periodic disturbance as time progresses, by observing the error in the previous period. Iterative Learning Control (ILC) is similar, it aims to converge to zero tracking error of system repeatedly performing the same task, and also adjusting the command to the feedback controller each repetition based on the error in the previous repetition. Compared to the conventional feedback control design methods, RC and ILC improve the performance over repetitions, and both aiming at zero tracking error in the real world instead of in a mathematical model. Linear Model Predictive Control (LMPC) normally does not aim for zero tracking error following a desired trajectory, but aims to minimize a quadratic cost function to the prediction horizon, and then apply the first control action. Then repeat the process each time step. The usual quadratic cost is a trade-off function between tracking accuracy and control effort and hence is not asking for zero error. It is also not specialized to periodic command or periodic disturbance as RC is, but does require that one knows the future desired command up to the prediction horizon.

The objective of this dissertation is to present various design schemes of improving the tracking performance in a control system based on ILC, RC and LMPC. The dissertation contains four major chapters. The first chapter studies the optimization of the design

parameters, in particular as related to measurement noise, and the need of a cutoff filter when dealing with actuator limitations, robustness to model error. The results aim to guide the user in tuning the design parameters available when creating a repetitive control system. In the second chapter, we investigate how ILC laws can be converted for use in RC to improve performance. And robustification by adding control penalty in cost function is compared to use a frequency cutoff filter. The third chapter develops a method to create desired trajectories with a zero tracking interval without involving an unstable inverse solution. An easily implementable feedback version is created to optimize the same cost every time step from the current measured position. An ILC algorithm is also created to iteratively learn to give local zero error in the real world while using an imperfect model. This approach also gives a method to apply ILC to endpoint problem without specifying an arbitrary trajectory to follow to reach the endpoint. This creates a method for ILC to apply to such problems without asking for accurate tracking of a somewhat arbitrary trajectory to accomplish learning to reach the desired endpoint. The last chapter outlines a set of uses for a stable inverse in control applications, including Linear Model Predictive Control (LMPC), and LMPC applied to Repetitive Control (RC-LMPC), and a generalized form of a one-step ahead control. An important characteristic is that this approach has the property of converging to zero tracking error in a small number of time steps, which is finite time convergence instead of asymptotic convergence as time tends to infinity.

CONTENTS

List of Figures

List of Tables

Acknowledgements

I am using this opportunity to express my gratitude to those who have helped me during my PhD study. It's their help and support that encourage me to face all the challenges and difficulties.

Firstly, I would like to express my sincere gratitude to my advisor Prof. Richard W. Longman for the continuous support of my study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

I also want to thank my family: my parents and my brother Jianxin Zhu for supporting me throughout my Ph.D program and my life in general. Their unconditional and endless love has accompanied me whenever I come across difficulties.

Last but not the least, my sincere thanks goes to Dr. Te Li, Dr. Yunde Shi, and Dr. Yao Li, who provided me an opportunity to know and join this team, and without their precious support it would not be possible to conduct this research, and I would also like to thank my labmate Xiaoqiang, Ae, Bing, Francesco and Henry. Their company made this long

journey much more fun and easier.

To my parents,

Mr. Jigen Zhu and Mrs. Yuefang Wang

This page intentionally left blank.

Chapter 1

Introduction

Motivation and Background

One of the main objectives of control system is to track a desired trajectory. A typical controller such as proportional, integral and derivative (PID) control, relies solely on the current tracking error observed in real time to generate corrective action. One problem with such a feedback system is that the controller continuously reacts to recurring errors as if they were completely new. Repetitive (RC)(References [1], [2], [3], [4], [5], [6], [7], [8], [9]) and iterative learning (ILC)[7], [10], [11], [12], on the other hand, make use of knowledge that the command is periodic or that the disturbance is periodic to ”learning” from the previous tracking errors, converging to zero tracking error.

The ILC problem considers that the control task is to perform a specific tracking command many times. Between each command application, the system is returned to the same initial condition, which is on the desired trajectory. The learning law simply adjusts the command to the feedback controller from one iteration to the next, in order to decrease

tracking error. Tracking error comes from several sources. First, the deterministic repeatable errors made in following general tracking commands. Second, there are deterministic disturbances that occur each time the same command is given. Third, there will always be some random disturbance errors. In repetitive control, the command to be executed is a periodic function of time. There may be deterministic disturbances that have the same period. For example, the period would be the same for gravity torque disturbance on a robot link performing a periodic motion in the work space. In repetitive control, there is no returning of the system to the same initial condition before the start of the next period, and thus transients can propagate across periods.

Model Predictive control (MPC) (Reference [13])is a control technique which uses internally the plant model to compute the system output predicted over a chosen time horizon into the future, and determines the optimal input to minimize a chosen cost function. The computations are made in real time applying the initial result, and then repeating the computation at the next times step. One of the nicest properties of linear MPC is that it can be tuned by adjusting the simple and intuitive parameters in quadratic cost function.

Thesis Outline

The thesis consists of five chapters. The main body of the original research work is divided into four major topics and included in Chapters 2 through 5. Chapter 1 offers introductory information about iterative learning, repetitive and model predictive control.

Chapter 2 studies the optimization of the design parameters in RC, incorporates physical considerations such as measurement noise, actuator saturation, robustness to model error.

Methods of producing a cutoff and of decreasing the size of the gains are considered. The results aim to guide the user in tuning the design parameters when creating RC control systems. Chapter 3 presents design approaches in RC problems using control laws generated for the ILC problem. The decreased learning rate at high frequencies provided by the ILC laws can be seen to improve robustness to model error when converted to be used in RC problems.

RC and ILC are examples of control laws that solve an inverse problem, in both cases by iterating with the real-world response. They can achieve zero error in the real-world model instead of our model of the world. But a more basic problem is to address the inverse problem for any given model, finding that input that will produce the desired output. In each of these cases, the inverse problem can very often be unstable, limiting the performance, and sometimes demonstrating the instability in application. In Chapters 4 and 5,we present recent results that develop methods of creating a new stable inverse solution are studied. Chapter 4 creates a method to use the stable inverse method locally, employing typical feedback control for tracking in much of the desired trajectory, and transitioning to high precision motion using the stable inverse theory based on one's system model for a desired high precision portion of the tracjectory. This is then extended to include ILC to locally learn to get zero tracking error in the high precision portion of the trajectory in the real world, eliminating the influence of imperfections in the model used for control design. ILC is normally a tracking control problem, but people often want to use it to address endpoint control problems. The transitioning to high precision tracking approach developed in this chapter creates a natural way to make ILC address such endpoint problems. Chapter 5 outlined a set of uses for the stable inverse, including LMPC, LMPC used for RC prob-

lems (RC-LMPC), and a generalized form of one-step ahead control. The fact that most discrete-time physical systems have an unstable inverse has prevented effective use of inverse ideas in control design. The presence of new ways to create stable inverses for such problems offers new opportunities. To within the accuracy of the model, the approach has the advantage of converging to zero tracking error in a small number of time steps, instead of convergence to zero error asymptotically as time goes to infinity.

Chapter 2

Incorporating Physical Considerations in the Design of Repetitive Controllers

This chapter studies the optimization of the design parameters. Because effective repetitive control designs eliminate all periodic error harmonics up to Nyquist frequency, the FIR compensator gains can be large. This is studied, in particular as related to measurement noise. Actuator limitations, robustness to model error, and noise can suggest that one should cut off the learning process at sufficiently high frequency. Methods of producing a cutoff, and of influencing the size of the gains are considered, including: adjustment of the sample rate, penalizing large gains in the FIR design, using a high frequency cutoff in the cost function for the FIR design, using a zero-phase low-pass filter of the repetitive control action, and combinations of these. These results aim to guide the user in tuning the design

parameters available when creating repetitive control systems.

2.1 Introduction

Repetitive control (RC) is a method of control that applies to situations where a feedback control system is subject to a periodic disturbance of known period, or is given a periodic command, or both (References [1], [2], [3], [4], [5], [6], [7], [8], [9]). This type of control can in theory converge to zero tracking error in each case. The simplest form of RC uses the error measured one period back, and changes the command this period by a gain times this error, aiming to correct the error. Mathematically it is iterating with the real world instead of a model. For frequency components having 180 degree phase lag through the feedback control system at some frequency, this algorithm will add to the error instead of decrease the error, so one needs to design a compensator to adjust phase. A periodic error has a fundamental frequency, but also has harmonics that can go up to Nyquist frequency. So RC initially aims to correct errors at all harmonics to Nyquist. This is completely different than usual feedback control system design that has a bandwidth, and above some frequency model errors are not important. Note that using the known periodic nature of the disturbance or command, RC is in theory able to correct such errors far above the feedback control system bandwidth, and this offers substantial improvement in the precision of control systems, achieved in software modification. But asking for zero error all the way to Nyquist pushes the design and hardware limits. Reference [9] offers a particularly effective method of designing compensators for repetitive control systems. Here we study this design method in detail, developing understanding of how to tune each of the design

parameters involved.

2.2 The Approach to Design Effective Repetitive Control Systems

The initial objective of RC is to converge to zero tracking error following a periodic command in the presence of a periodic disturbance both having the same known period of p sample time steps. An important special case is that of a constant command in the presence of a periodic disturbance. The usual repetitive control system structure uses an existing feedback control system, and the repetitive control is an extra loop around this that adjusts the command to the feedback system. There are some variations to this structure (see e.g. Reference [8]) but the basic mathematics remains very similar. The ideal RC law is the inverse of the feedback control system discrete time transfer function, but this is very often unstable and cannot be used. Here we study RC design using the very effective approach from [9] (see also [8] which uses the inverse of the feedback control system steady state frequency response, generated as a finite impulse response (FIR) function, and this cannot be unstable.

Figure 2.1 shows the structure considered here which has a continuous time feedback control system fed by a zero order hold. Similar results would apply if it were a digital control system with a continuous time plant. The z-transform of the desired output is $Y_D(z)$, the repetitive controller $R(z)$ adjusts the command $U(z)$ to the feedback control system, which can be subject to a periodic disturbance $\bar{V}(s)$ shown entering at the usual

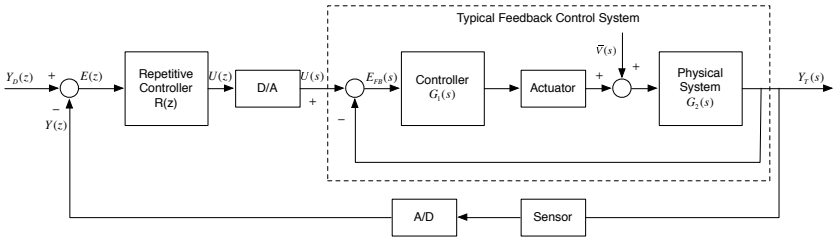

Figure 2.1: Block diagram of repetitive control systems considered

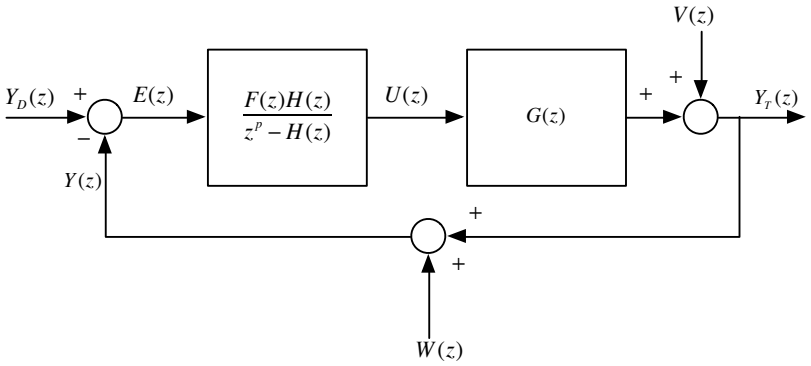

Figure 2.2: Digital RC system equivalent to Figure 2.1 with equivalent output disturbance and measurement noise

location. Figure 2.2 introduced the equivalent digital closed loop transfer function $G(z)$ of the feedback control system, with an equivalent additive periodic disturbance $V(z)$ on the feedback system output. Introduced in this figure is measurement noise $W(z)$ added to the sensor measurements used by the repetitive controller. Again, to examine the essence of the problem without unnecessary complexity, we do not include measurement noise to the feedback control system model. The actual output of the RC system is $Y_T(z)$ while the measured output corrupted by noise is $Y(z)$.

The simplest form of repetitive control law $u(k) = u(k-p) + \phi e(k-p+1)$ looks at the error (desired output minus measured output) one period back, but shifted one step forward assuming the time delay through the feedback control system is one time step. The

new command at the current time step is the command one period back plus the gain ϕ times this error. This is analogous to creating a discrete time integration of the errors for each time step of the period, and like integral control applied to a constant disturbance, it either makes the associated error go to zero, or the control system goes unstable. This is generalized in z-transfer function form

$$U(z) = z^p \left[U(z) + F(z)E(z)\right]; \quad U(z) = \left[\frac{F(z)}{z^p - 1}\right] E(z) \tag{2.1}$$

by introducing a compensator $F(z)$ in place of ϕz (the $H(z)$ in Figure 2.2 is set to unity until later). Block diagram algebra produces what can be interpreted as a difference equation

$$\left[z^p - (1 - G(z)F(z))\right] E(z) = (z^p - 1)\left[Y_D(z) - V(z) - W(z)\right] \tag{2.2}$$

We consider that both $Y_D(z)$ and $V(z)$ are periodic with p time steps so the right hand side becomes zero except for the influence of the measurement noise $W(z)$. Stability is a property of the homogeneous equation which can be rewritten in form

$$z^p E(z) = (1 - G(z)F(z))\, E(z) \tag{2.3}$$

The factor $\left[1 - G(e^{i\omega T})F(e^{i\omega T})\right]$ looks like a frequency transfer function from the error in one period to the error in the next period. Requiring that it's magnitude response be less than unity for all frequencies up through Nyquist

$$|1 - G(z)F(z)| < 1 \quad \forall z = e^{i\omega T} \tag{2.4}$$

would suggest monotonic decay of the amplitude of every frequency component of the error, which indicates asymptotic stability. This is not rigorous because, frequency response is a steady state property, and if the system is stable then the error on the right and on the left are both zero. Instead, one can then claim that if the learning is sufficiently slow to consider it as quasi-steady state, then stability is indicated. Reference [7] and Reference [8] show that by rigorous arguments that the inequality in Equation 2.4 is a necessary and sufficient condition for asymptotic stability for all possible periods p. Reference [8] also show that because p is usually a large number of time steps, $\left[1 - G(e^{i\omega T})F(e^{i\omega T})\right]$ is in fact a very good estimate of the decay of error for each frequency from one period to the next, even when the decay is very fast the quasi-static assumption is not normally a limiting factor.

Equation 2.4 suggests that a particularly good compensator would be $F(z)$ set equal to $G^{-1}(z)$. As mentioned above, the inverse of perhaps most discrete time equivalents of continuous time transfer functions is unstable, and this precludes using this design. Instead we design a finite frequency response compensator (FIR), $F(z)$, that mimics that the steady state frequency response of $G^{-1}(z)$. The compensator takes the form

$$\begin{aligned} F(z) &= \frac{a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_n}{z^{n-m}} \\ &= a_1 z^{m-1} + a_2 z^{m-2} + \cdots + a_m z^0 + \cdots + a_{n-1} z^{n-m-1} + a_n z^{n-m} \end{aligned} \tag{2.5}$$

which represents a linear combination of n errors from the previous period, a_m is the coefficient of the error one period back, $m-1$ errors are future to the time step one period back, and $n - m$ are past time steps from the step one period back. The coefficients are chosen

to minimize the cost function

$$J = \sum_{j=1}^{N} [1 - G(e^{i\omega_j T})F(e^{i\omega_j T})][1 - G(e^{i\omega_j T})F(e^{i\omega_j T}))]^T \tag{2.6}$$

The ω_j form a discrete set of frequencies from zero up to Nyquist frequency, and the upper limit on the summation N can be chosen to pick the full set, or it can be chosen to cut off the summation before reaching Nyquist frequency.

The numerical examples presented in this chapter use the following continuous time system

$$G(s) = \left(\frac{a}{s+a}\right)\left(\frac{\omega_1^2}{s^2 + 2\zeta\omega_1 s + \omega_1^2}\right) \qquad a = 8.8, \omega_1 = 37, \zeta = 0.5 \tag{2.7}$$

2.3 Performance of Effective RC Approach

The magnitude frequency response of the FIR compensator $F(z)$ for the third order system above is shown in Figure 2.3 along with a plot of the magnitude frequnecy response of $G^{-1}(z)$. Using only 12 gains produces a compensator that is indistinguishable from $G^{-1}(z)$ to graphical accuracy. Figure 2.4 plots the left hand side of Equation 2.4 with a maximum value of approximately 3.5×10^{-3}, which guarantees stability of the RC systems, and suggests very fast convergence. To see the convergence, consider a disturbance $V(z)$ that is a one Hertz sine wave of amplitude one, with period $p = 200$ time steps, and the desired output is zero. For the initial run before tuning on the RC system, the command to the feedback control system is zero and the true output $Y_T(z)$ is then the output disturbance

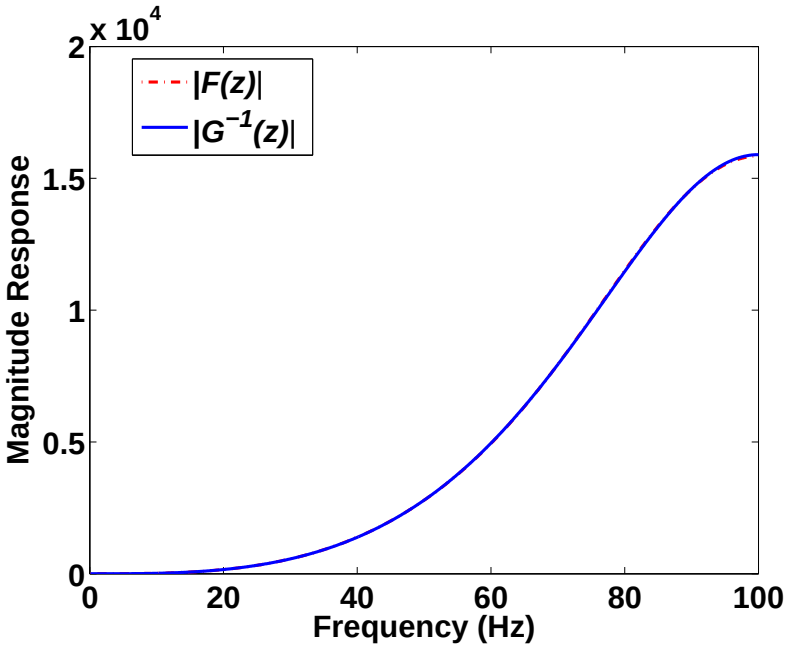

Figure 2.3: Magnitude plot vs. frequency for $G^{-1}(z)$ and for 12 gains FIR fit $F(z)$

Table 2.1: Standard deviation of measurement noise and resulting true output

Sample Rate	S.D. $W(z)$	S.D. $Y_T(z)$
200 Hz	0.9906	0.9793
400 Hz	0.9906	0.9809

$V(z)$. This is called repetition 1 in Figure 2.5. We see that the RC root mean square (RMS) error for each repetition decays very fast with iterations, and at the 8^{th} iteration the error is 1.4×10^{-14}. This is essentially the final error level reached by the computer simulation and corresponds to a numerical zero. This is very good performance in the noise free case.

Now we consider RC with measurements corrupted by zero mean white Gaussian noise $W(z)$ generated with standard deviation of unity. Using sample rate 200 Hz and examining 100 periods of data after convergence, and 400 Hz with the same number of samples. The standard deviation of the noise and the output samples are given in Table 2.1. From this one might consider that the RC system is also very well behaved in the presence of noise.

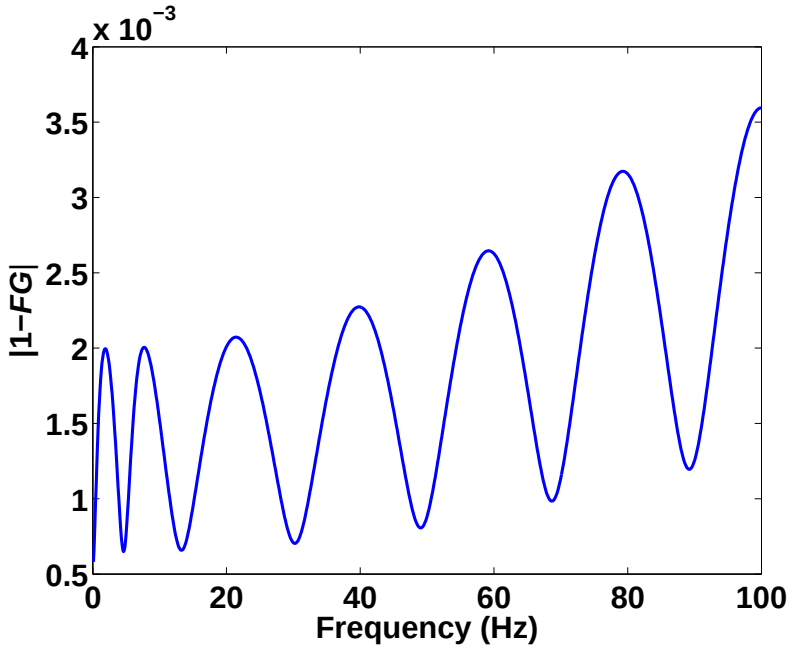

Figure 2.4: Magnitude response of $|1 - F(z)G(z)|$ corresponding to Figure 2.3

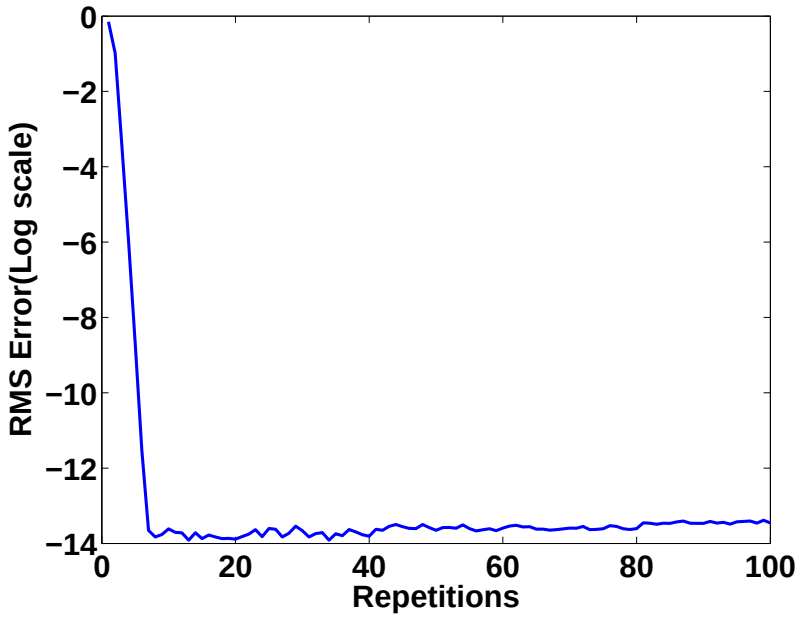

Figure 2.5: RC system RMS error per repetition without noise

2.4 Possible Concerns

Actuator Requirements to Control all the Way to Nyquist

Routine feedback control system design if often characterized by the bandwidth, the control system does a reasonable job of executing commands, or command components above this number. RC is designed to produce zero tracking error of periodic commands, or in the presence of periodic disturbances of a given period. This is includes DC (or zero frequency), the fundamental of the given period, and all harmonics up to Nyquist. The closed loop transfer function of the feedback control system $G(z)$, like all such systems has a frequency response that decays with frequency. The compensator $F(z)$ is a very good approximation $G^{-1}(z)$ (as seen in Figure 2.3 for 200 Hz sample rate), which means that it asks for very large commands to the feedback control system when asked to eliminate a harmonic near Nyquist frequency. Table 2.2 gives the maximum amplification factor in $F(z)$ associated with Nyquist frequency, and also shows the amplification in the actual $G^{-1}(z)$ and the difference between these two indicating how accurate the 12 gain FIR approximation of $G^{-1}(z)$ is. To interpret these numbers, consider an error of amplitude unity near Nyquist frequency of 500 Hz for the sample rate of 1000 Hz. The command needed to cancel this error has amplitude 1.9784733×10^6. This is purely a property of the physical system, and is independent of the choice of RC law to accomplish the zero error. In the configuration of Figure 2.1 that has a continuous time feedback control system, the table talks about the size of the command given to the system for zero error. Similar results would apply if the feedback control system were digital, and one creates the required output of its digital control law being sent to the actuator through a zero order hold. Clearly this amplitude can

Table 2.2: Maximum amplification of the compensator $F(z)$ and of the system inverse $G^{-1}(z)$ at Nyquist frequency

Sample Rate	$max\|F(e^{i\omega T})\|$	$max\|G^{-1}(e^{i\omega T})\|$	Difference
200 Hz	1.584×10^4	1.59×10^4	60
400 Hz	1.266475×10^5	1.27095296×10^5	447.8
1000 Hz	1.9784733×10^6	1.985441569×10^6	6968.3

be prohibitive.

Noise Amplification

The previous section was concerned with the large amplitude needed to fix errors at high frequencies, but this same large amplification applies to measurement noise as well. The top part of Table 2.3 considers the same white zero mean Gaussian noise with standard deviation of unity as before. Instead of looking at the noise effect on the true output we look at the noise influence on the command given the feedback control system. As before the command is zero, so the command is only responding to noise. The standard deviation of the command is given when the RC system is running. Consider that the feedback control system has unity DC gain, so that any frequency near DC is corrected by a deterministic signal of amplitude near unity. But correcting such a signal is subject to the influence of the measurement noise, which produces a standard deviation around this deterministic value that is $7,268.8$ for 200 Hz sample rate, or $58,618$ for 400 Hz sample rate. Such amplification of noise could be a problem even though when the signal goes through the feedback control system $G(z)$ it gets attenuated back to approximately unity standard deviation. Also shown in the last column is the standard deviation of just the noise going through the compensator $F(z)W(z)$. Note that since $F(z)$ produces a very good approximation of the frequency response of the feedback

Table 2.3: Standard deviation of the command due to white noise and pink noise

	S.D.$W(z)$	S.D. $U(z)$ RC running	S.D. $F(z)W(z)$
WHITE			
200 Hz	0.9906	7,268.8	7,314
400 Hz	0.9906	58,618	
PINK			
200 Hz	0.2392	556.19	559
400 Hz	0.2392	4,486.5	
1000 Hz	0.2392	70,451	

control system, this amplification factor is purely a result of designing the compensator to be the inverse of the system frequency response. In this respect it is again a system property.

One might question to what extent white noise is a reasonable approximation of the noise that might actually be encountered in hardware application. White noise assumes uniform amplitude at all frequencies from zero to Nyquist, but noise in applications is likely to decay substantially with frequency. Pink noise has a power spectral density that is inversely proportional to frequency. Starting with the same $20,000$ sample white noise history, an algorithm is used to convert it to pink noise, and the frequency spectrum from discrete Fourier transforms for each is compared in Figure 2.6. The pink noise and white noise amplitudes intersect at one Hertz, and the pink noise decays according to the reciprocal of the square root of the frequency. The bottom of Table 2.3 gives the corresponding results for standard deviation of the command and of $F(z)W(z)$. The command standard deviation is very much reduced, but the value is still rather large.

Quantification

The control system designer picks the zero-order hold analog to digital converter that feeds the command to the continuous time feedback control system. This hardware will need the

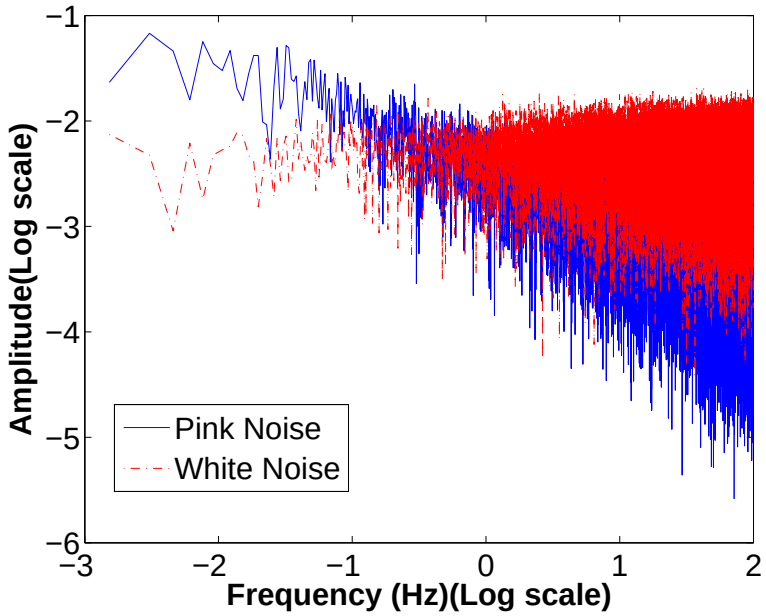

Figure 2.6: Frequency content of white noise and pink noise sample

specification of the full-scale range, and the total number of discrete levels. The designer might pick the full-scale range large enough that most of the noise distribution is included. But since this number may be quite large as indicated in the previous section, the result might be that the discrete levels are far apart, and there is considerable quantization error. Reference [14] studies the influence of quantization on the final error level in learning and repetitive control.

The actuator in the feedback control system will have its own actuator limits, and too large a full-scale range on the command may result is a saturation nonlinearity of the actuator. If instead the feedback control system is digital, and has a digital controller whose output goes through the digital to analog converter to feed the actuator, then the full-scale range can be chosen directly based on actuator limits. In either case there is potential for either degradation of performance due to quantization or to actuator saturation.

Noise truncation and bias

The effect of truncating noise can be to introduce bias into the converged RC system, making it fail to converge to zero error for periodic commands and disturbances. This is illustrated by the following simulation. We set the desired output to a constant value of 35. The standard deviation of the noise $W(z)$ is set to 0.002 which results in a standard deviation of command about 35 equal to 14.72. Then we set the upper limit and low limit of the full-scale range equal to ± 42.36. Therefore, the upper limit is the desired output of 35 plus one half of a standard deviation due to noise. The simulation is run truncating $u(k)$ to the full-scale range whenever the repetitive control law asks for something that exceeds the full-scale range. The deterministic system has no error in following a constant command set to 35. With noise added and truncated when necessary, the steady state average value of the command is 32.0761. And there is a steady state error or bias of 2.92017 in following this "periodic" command. Therefore, noise going beyond the full-scale range, or going beyond a hard actuator limit, will generally result in a bias in the converged result. Remember that the noise level in the output is small, of a similar value to the measurement noise level itself, even when the system can perform the desired output command. Here there is substantial bias in the converged result caused by a low noise level.

A control practitioner might say, if the measurements are noise, why don't you run them through a low pass filter before you use them in the repetitive control law. Of course, this can decrease quantization or bias problem. but now the error signal $E(z) = Y_D(z) - Y(z)$ used by the repetitive controller in Equation 2.1 is replaced by $E_F(z) = Y_D(z) - Y_F(z)$ where $Y_F(z) = F_2(z)Y(z)$ is the measurement filtered through low pass filter $F_2(z)$. Equations

2.2, 2.3 and 2.4 become

$$\left[z^p - (1 - G(z)F(z)F_2(z))\right] E_F(z) = (z^p - 1)\left[Y_D(z) - F_2(z)V(z) - F_2(z)W(z)\right] \tag{2.8}$$

$$z^p E_F(z) = (1 - G(z)F(z)F_2(z))E_F(z) \tag{2.9}$$

$$|1 - G(z)F(z)F_2(z)| < 1\forall \quad z = e^{i\omega T} \tag{2.10}$$

Since $V(z)$ is periodic with period p in steady state, so is $F_2(z)V(z)$, and again the right hand side of difference Equation 2.8 contains only the forcing function from the noise, and this time the filer has made the forcing function smaller, and the associated particular solution should also be smaller. However, the solution to the homogeneous equation, if stable, satisfies Equation 2.9, and now it is the filtered error that converges to zero, not the actual error. A causal filter $F_2(z)$ will introduce phase lag and amplitude attenuation, so it is clear that one is no longer aiming for zero error. In addition, stability is determined by Equation 2.10. Of cause one could ask for $F(z)F_2(z)$ to mimic $G^{-1}(z)$, but then you defeat the purpose of the filter by amplifying the noise to cancel the filter. And if one designs $F(z)$ as before, the RC system is most likely unstable. A first order continuous time system puts in a phase lag of 90 degree at infinity, and the digital version probably does 180 degrees at Nyquist, and wither amount of extra phase lag is enough to violate Equation 2.10 and destabilize the system. To avoid these issues one can consider using a zero phase noncausal filter on the data from the previous period, which we consider later as applying not just to the measured error but to the total control action, the filter $H(z)$ in Figure 2.2.

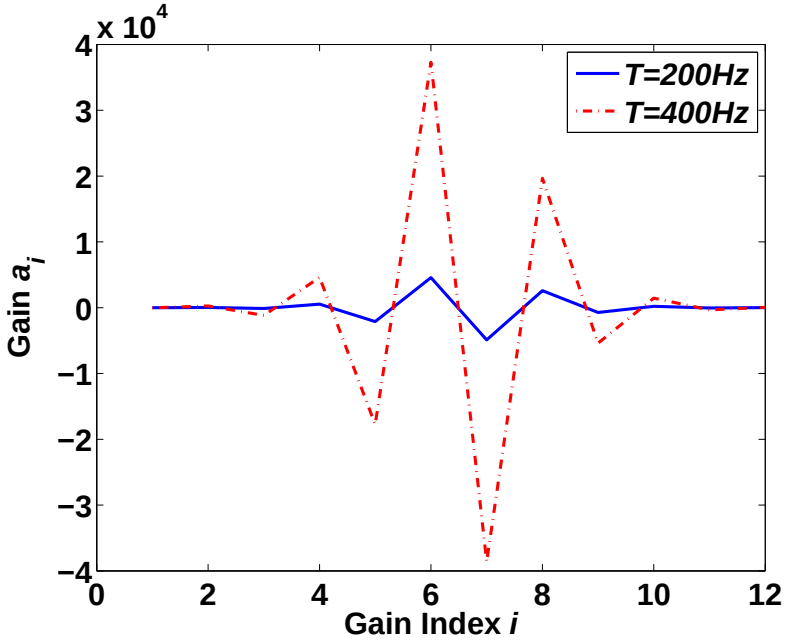

Figure 2.7: The 12 gains of $F(z)$ for 200 Hz and 400 Hz sample rate

Table 2.4: Two largest gains in magnitude in $F(z)$

	a_7	a_6
200 Hz	-4897	4564
400 Hz	-38620	37290

Possible numerical issues with large controller gains

In addition to large amplification of noise, the RC design may lose significant digits during the numerical computation. Compensator $F(z)$ uses 12 gains a_i, as indicated in Equation 2.5 and plotted in Figure 2.7 for both 200 Hz and 400 Hz sample rate. Note that the largest gain in magnitude is a_7 (which is a negative gain), while the second largest gain a_6 applies to one time step forward in time. These two gains are similar in magnitude but opposite in sign, as shown in Table 2.4. Figure2.8 plots the absolute value of these two gains as a function of sample rate, and the two plots are hard to distinguish to graphical accuracy.

We comment that one should expect something similar to this if one is to correct errors

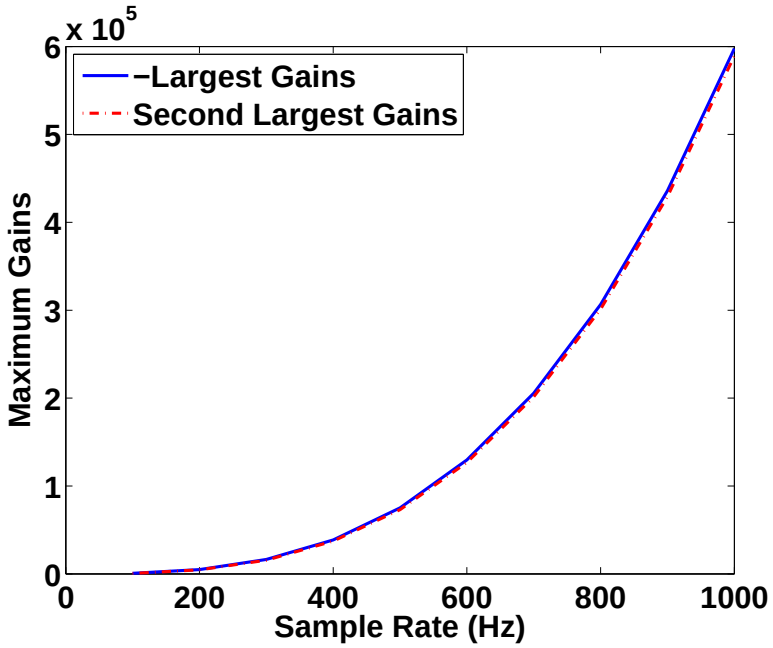

Figure 2.8: The absolute value of the largest fain and the value of the second largest gain of $F(z)$ versus sample rate

at each time step for frequencies near Nyquist. At Nyquist frequency there are two samples per period and the data is not able to determine the phase or the amplitude of the sinusoid. Near Nyquist this becomes an ill-conditioned problem. Suppose one obtains samples of a sinusoid at successive time steps k

$$Msin(\omega kT + \theta) = M[(sin\theta)cos(\omega kT) + (cos\theta)sin(\omega kT)] \tag{2.11}$$

At Nyquist frequency, $\omega = \pi/T$, and $sin(\omega kT) = 0$ for all k, and $cos(\omega kT)$ is $+1$ for even k and -1 for odd k. Hence one can only know the product $Msin\theta$ and cannot determine M or θ. If ω is near Nyquist then knowledge of two successive values $y(k)$ and $y(k+1)$

of the left hand side of Equation 2.11 allows you to solve for M and θ from

$$\begin{bmatrix} y(k) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} cos(\omega kT) & sin(\omega kT) \\ cos(\omega(k+1)T) & sin(\omega(k+1)T) \end{bmatrix} \begin{bmatrix} Msin\theta \\ Mcos\theta \end{bmatrix} \tag{2.12}$$

At Nyquist the coefficient matrix is singular, but near Nyquist it will be ill conditioned, and one expects that the successive nearly equal gains of opposite sign are a manifestation of this ill conditioning. Small changes in the measured values on the left can be associated with large changes in the associated magnitude and/or phase of the signal measured. The computation of the product of the compensator operating on the 12 measured errors is likely made in a minicomputer or micro processor that can have a substantial word length, which limits the computation errors. But the measured errors come from physical sensor hardware, and the number of digits measured accurately is usually rather limited. Consider a situation where the measurement sensor operates on a scale from -10.0 to $+10.0$ units, so that each measurement is accurate to ± 0.05. If the gains a_i have a larger range of magnitudes than the 3 digits accuracy of each error, then the computation of the sum of 12 terms can have numerical inaccuracy.

Now return to the concept of amplification of the noise corrupting measurements, this time seen in terms of the gains involved, instead of frequency response. The sum of the 12 gains is always unity, because a continuous time system with a DC gain of unity also has a DC gain of unity when fed by a constant function going through a zero order hold. Suppose that all 12 error measurements were corrupted by the same positive constant value α, for example $+0.05$. Then the command computed from $F(z)E(z)$ would be corrupted by the

amount

$$\left(\sum_{i=1}^{12} a_i\right)\alpha = \alpha \tag{2.13}$$

Now suppose that each measurement is corrupted by a constant with magnitude α but with the sign made equal to the sign of the corresponding a_i, alternating sign every time step. Then the influence of this error of constant magnitude each time step will corrupt the computed command to the feedback control system by

$$\left(\sum_{i=1}^{12} a_i\right)|\alpha| = \begin{cases} 15,842 \quad |\alpha| \qquad for 200Hz \\ 126,647 \quad |\alpha| \qquad for 400Hz \\ 1,978,473 \quad |\alpha| \qquad for 1000Hz \end{cases} \tag{2.14}$$

The need for a frequency cutoff/limiting the size of gains

The previous section identified two basic issues. The first is the amount of control or actuator effort needed to correct errors in physical systems at high frequency. This is purely a physical hardware issue, and is not specific to any particular repetitive control system design. The second issue is both the size of the gains and the range of sizes of the gains in the compensators designed to mimic the inverse of the steady state frequency response of the system. One can directly address the size of the gains by modifying the cost function in Equation 2.6 to include a penalty on the sum of the squares of the gains with a weight factor W_a

$$J = \sum_{j=1}^{N} \left[1 - G(e^{i\omega T})F(e^{i\omega T})\right]\left[1 - G(e^{i\omega T})F(e^{i\omega T})\right]^T + a^T W_a a; \quad a^T = [a_1 \quad a_2 \quad \dots \quad a_n] \tag{2.15}$$

This was suggested in the original publication on the method, Reference [9]. The first issue argues for having a cutoff of the repetitive control action above some chosen frequency, and we consider a cutoff filter$H(z)$. The cutoff may also be an indirect way to address the size of gains issues. We can list a series of interrelated reasons to want to stop the learning, i.e. the convergence to zero error process, for error components above some cutoff frequency:

- Gain size: Possible benefit of reducing the size of the gains, and the range of sizes of the gains.
- Noise: Limit the amplification of noise, to eliminate bias from noise truncation, reduce the full-scale range needed on the digital to analog converters, reduce quantization error.
- Actuator limits: Correcting errors at too high a frequency can require control actions that the actuator cannot deliver at such a frequency, and result in saturation.
- Converter: As discussed with noise, simply correcting errors at very high frequencies can ask for large full-scale range with possible quantization effects.
- Be kind to the hardware: It may be important to not try to eliminate errors at too high a frequency, in order to be kind to the hardware. References [7] and [8] and a reference therein, discuss a case where the RC was too good, it eliminated the full spectrum of error to Nyquist, but the hardware was making so much noise we figured we were wearing out the equipment. The decision was not to correct for the high frequency error due to tooth meshing in a gearing system.
- Energy: Correcting errors at very high frequency can require large force, or torque,

or energy, or power, or large fuel expenditure.

- Needed frequency range: If the periodic command of interest, and the periodic disturbances to the system, do not have significant high frequency content, then there is no need to ask to correct errors at high frequencies.
- Stability robustness: Perhaps the most important reason to limit the learning to lower frequencies, is stability robustness to model errors. In order to satisfy the stability condition Equation 2.4 all the way to Nyquist, one needs $F(z)$ to mimic the true $G^{-1}(z)$ with a phase error that must be less than $\pm 90°$.Failing to have one pole at high frequency in your model is enough to destabilize the learning process. One must cut the learning when the model is too poor to learn.
- Fidelity of signal representation: People often want to run digital control systems at as fast a sample rate as possible, because faster sampling gives a closer representation to the true continuous time signals, e.g. the disturbance. The same lower frequency disturbance is better canceled by a control system with its zero order hold running at a faster sample rate. But fast sample rates emphasizes many of the above needs for cutting off the learning above a chosen frequency.

In this list there are some natural cutoff frequencies. Concerning the need, the command and the disturbance may not have significant frequency content above some value. The choice of sample rate determines Nyquist frequency which is a special kind of cutoff. The RC law operates on a feedback control system, and every feedback control system has a bandwidth which is a form of frequency cutoff. Very often the control system designer is fighting to get the bandwidth up, and issues of stability of the feedback system are limiting factors. There

are other applications where the feedback control system designer intentionally restricts the bandwidth so as not to excite the first resonance frequency in a system, and as a result not to have to be clever in dealing with the extra dynamics. For feedback control design one is con-strained by stability issues. Fortunately, RC can address periodic errors far above the band-width of the feedback controller, and theoretically completely eliminate them. The limitations are model accuracy for stability, and actuator capability.

2.5 Parameter choices in the design of RC systems

The RC designer has various choices to make. He must pick the values of n and m for the compensator. We expect that a choice that makes the cost function very small, will result in large gains, but for the 3rd order system considered here the minimum number of gains for stability is 3. According to Reference [8] a stabilizing choice is $n = m = 3$, and the gains for $z^2, \quad z^1, \quad z^0$ are not particularly large, equal to 247.96, 488.7, 242.28 respectively. The price one pays for this is slow learning, i.e. the left hand side of the inequality in Equation 2.4 may be far from zero, but still not violate the inequality.

In addition to these two choices, the designer must pick the sample time interval T, and he can introduce the weight W_a in Equation 2.15. He can pick an upper limit frequency, ω_N, on the cost function in Equation 2.6 or 2.15 that is smaller than Nyquist frequency to indicate what part of the frequency spectrum is important. And the designer can modify the RC law in Equation 2.1 to include one or both of the following generalizations

$$U(z) = z^{-p}H(z)[U(z) + \phi F(z)E(z)]; \qquad U(z) = \left[\frac{\phi F(z)H(z)}{z^p - H(z)}\right] E(z) \tag{2.16}$$

The introduction of an overall gain ϕ multiplying $F(z)$ directly multiplies all gains in $F(z)$ by this factor and can be used to reduce these gains, and do so at the expense of slower learning at all frequencies from one repetition to the next. The $H(z)$ is a zero-phase low-pass filter that cuts all frequency content above the cutoff in the command going to the feedback controller, so that errors above the cutoff are being ignored. We now examine these different options.

Using a cost function weight W_a on size of RC gains

First we consider the use of the penalty term W_a on the size of the RC compensator gains as introduced in Equation 2.15. Figure 2.9 shows the substantial resulting change in the size of the gains when a very small weight $W_a = 0.00001$ is used, and Figure 2.10 gives the magnitude frequency response of the resulting $F(z)$, The largest gain has been reduced in magnitude very substantially from 4897 to 938 using this small weight. Of course, the learning process is slower. Suppose a cutoff of the learning is defined as the frequency when the radial distance from $+1$ to the plot of $G(z)F(z)$, $z = e^{i\omega T}$, equals 0.95. Figure 2.11 from Reference [15] gives this value as a function of W_a, and we see that this penalty function is making the learning at high frequencies become slow, and is functioning rather like a cutoff of the learning process at high frequencies.One may not want to push too hard at reducing the gain size in this manner. Figure [16] shows the plot of $G(z)F(z)$, $z = e^{i\omega T}$, near the near the origin for 200Hz sample rate and weight$W_a = 0.001$, and the plot appears stable, remaining inside the unit circle centered at $+1$. However, when the weight is increased to $W_a = 0.005$ the corresponding plot is given in Figure 2.13 and is

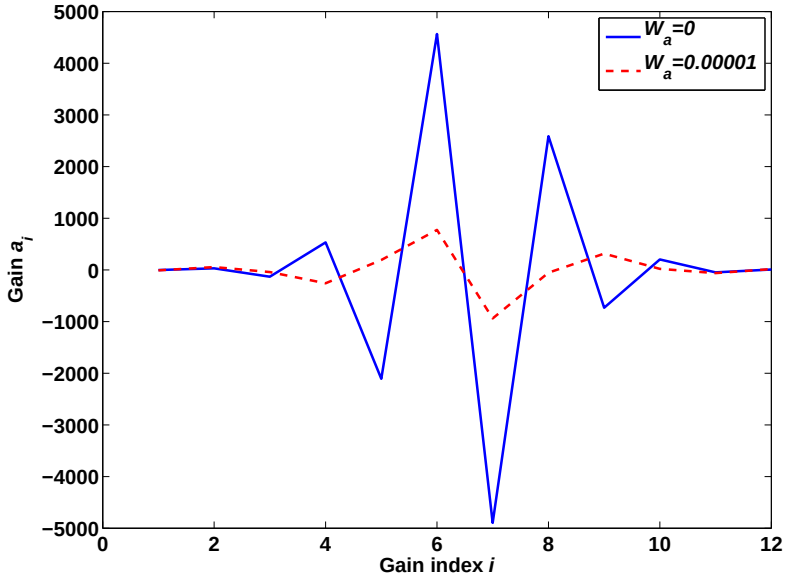

Figure 2.9: The 12 gains without penalty W_a in cost function J and with a small penalty

unstable. Figure 2.14 plots the value of the right hand side of inequality Equation 2.4 and the system goes unstable some-where around 40 Hz. Figure 2.15 shows that the RC system goes unstable when one goes back to $W_a = 0.001$ but increases the sample rate to 400Hz. Of course, if it goes unstable above a cutoff frequency that one wants to use for some other purpose, then stability is established by using the cutoff filter $H(z)$in the next section.

A Cutoff Filter $H(z)$

Using the cutoff filter as introduced in the RC law Equation 2.16, Equation 2.2, 2.3 and 2.4 become

$$[z^p - H(z)(1 - \phi G(z)F(z))]E(z) = (z^p - H(z))[Y_D(z) - V(z) - W(z)] \tag{2.17}$$

$$z^p E(z) = H(z)(1 - \phi G(z)F(z))E(z) \tag{2.18}$$

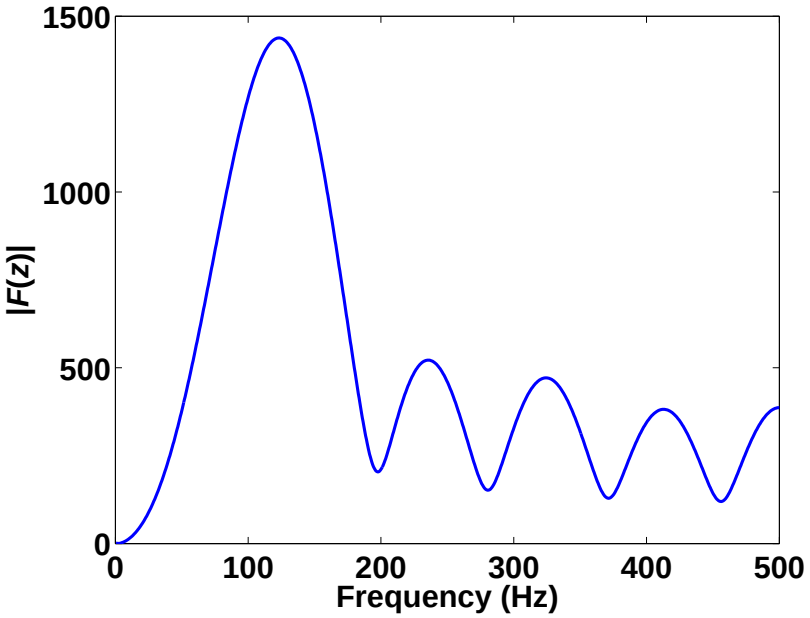

Figure 2.10: Magnitude frequency response of $F(z)$ using 1000Hz sample rate $W_a = 0.00001$

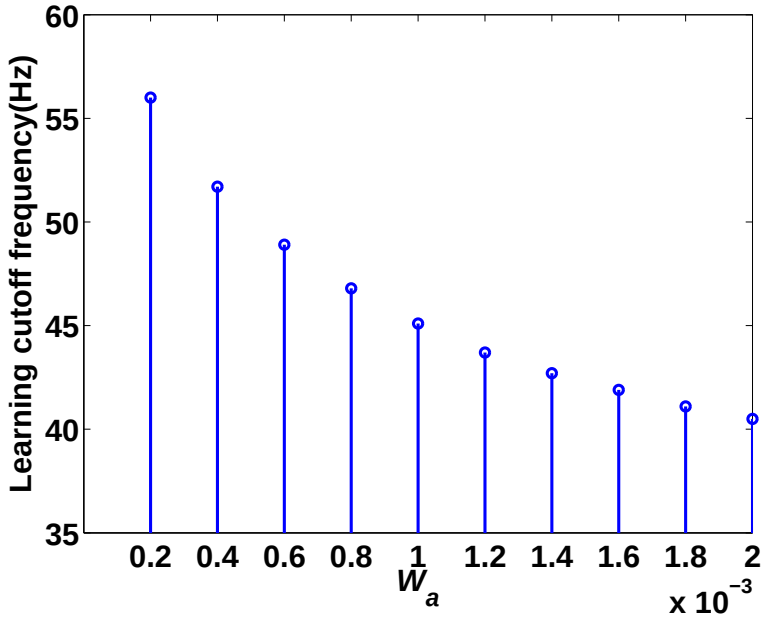

Figure 2.11: The effective cutoff frequency associated with different weights W_a

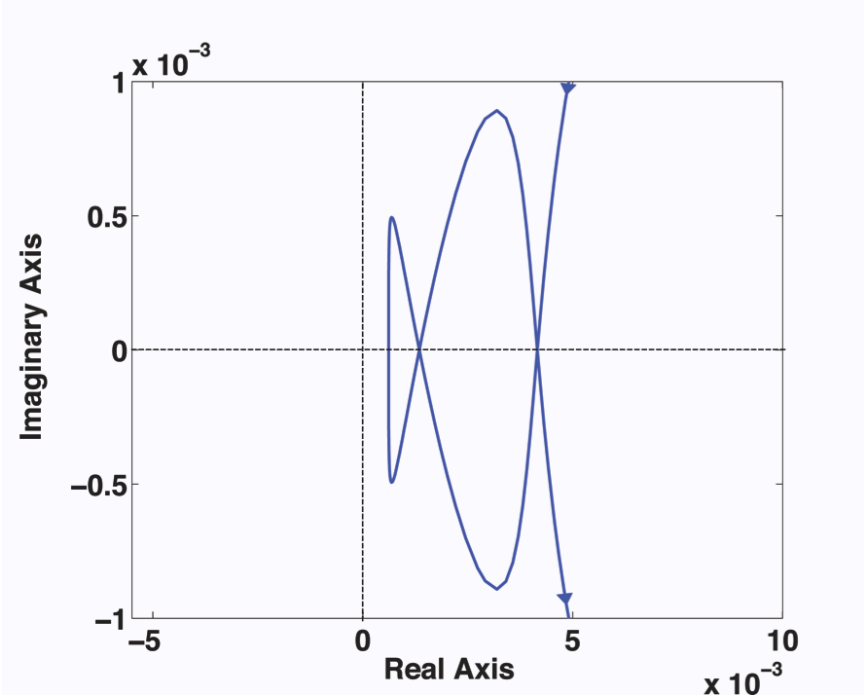

Figure 2.12: Frequency plot of $G(z)F(z)$ associated with $W_a = 0.001$ and 200 Hz sample rate

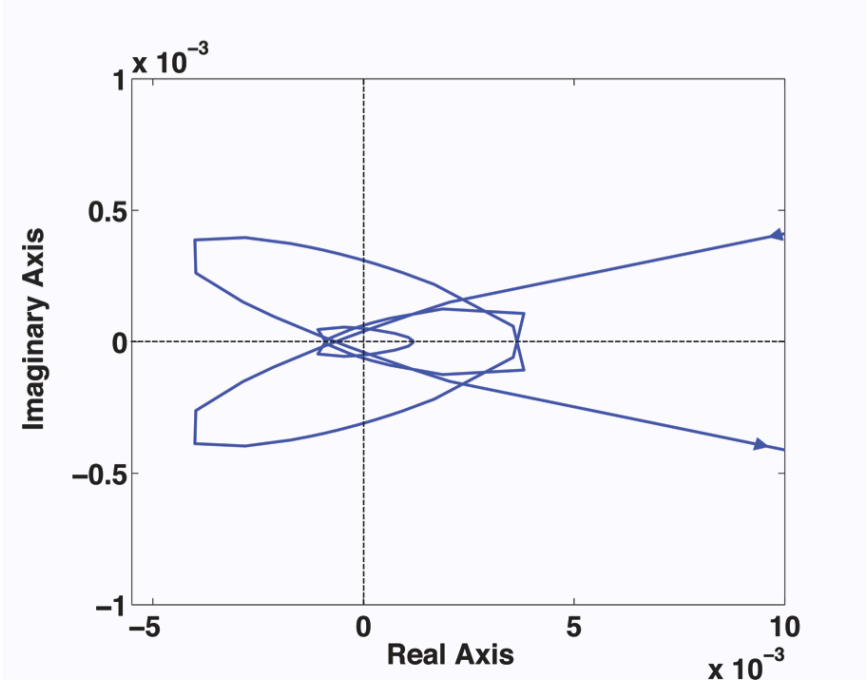

Figure 2.13: Frequency plot of $G(z)F(z)$ associated with $W_a = 0.005$ and 200 Hz sample rate

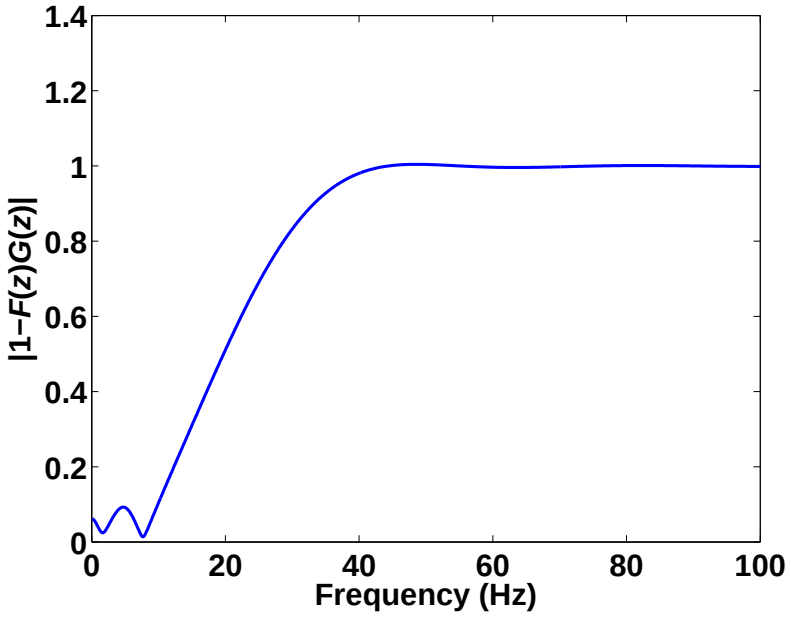

Figure 2.14: Plot of $|1 - F(z)G(z)|$ corresponding to Figure 2.13

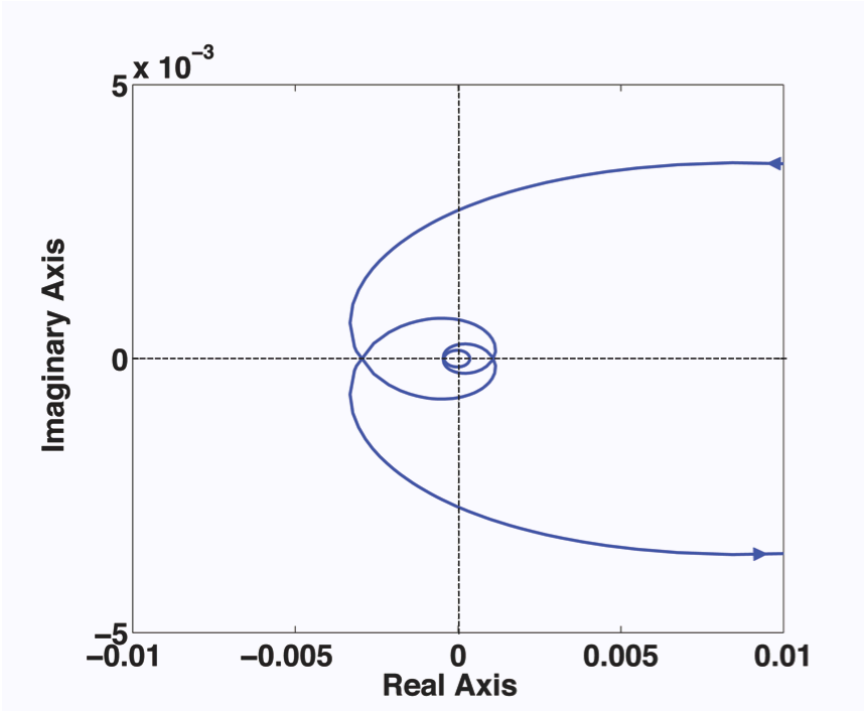

Figure 2.15: Frequency plot of $G(z)F(z)$ associated with $W_a = 0.001$ and 400 Hz sample rate

$$|H(z)[1-\phi G(z)F(z)]| < 1 \quad \forall \quad z = e^{i\omega T} \tag{2.19}$$

From the first of these equations it is clear that this filter should be zero phase filter that aims to equal unity below the cutoff in the passband. Then one aims for $(z^p - H(z))$ to equal $z^p - 1$ in the passband making the forcing function $Y_D(z) - V(z)$ equal to zero in this frequency range. If there were phase change as in a nonzero-phase filter there would be a forcing function that produces error. One asks for zero in the stopband, and improved performance is obtained if a transition band is introduced before the start of the stopband, then in the stopband $z^p - H(z) = z^p$, and there is a forcing function and associated particular solution associated with $Y_D(z) - V(z)$, producing error in this unaddressed frequency range. Note that in the stopband ideally the particular solution for $E(z)$ associated with the noise $W(z)$ forcing function is equal to $W(z)$, and is not amplified. Reference [16] introduces such a zero phase filter satisfying

$$H(z) = \sum_{k=-n}^{n} a_k z^k \tag{2.20}$$

$$J_H = \alpha \sum_{j=0}^{j_p} [1 - H(e^{i\omega_j T})][1 - H(e^{i\omega_j T})]^\star + \sum_{j=j_s}^{N-1} [H(e^{i\omega_j T})][H(e^{i\omega_j T})]^\star \tag{2.21}$$

where α is a weight factor to adjust the relative importance of stopband vs. passband, and the summation limits allow for a transition band. This makes a linear problem for solution of the odd number of filter weights, here we use 51 weights or gains. Reference [17] makes this into a quadratic programming problem with inequality constraints that ensure that the filter does not go above unity magnitude in the passband. If the cutoff frequency is being chosen for robustness to model error, one can tune the cutoff with hardware experiments. Using this constraint helps one satisfy the stability condition Equation 2.19 up to as high a frequency

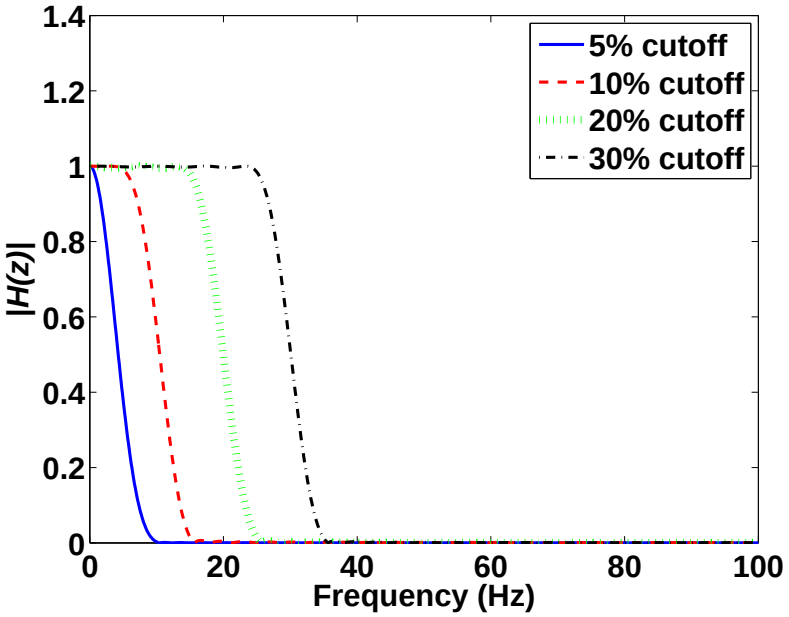

Figure 2.16: Zero phase filter designs for different percent Nyquist cutoffs

as possible. The numerical results reported here use the filter design from Reference [17]. It is also possible to not ask for zero error in the stopband, but rather just enough attenuation to satisfy the stabil-ity requirement in Equation 2.19, and this is studied in Reference [18]. Figure 2.16 gives the magnitude performance of the 51 gain design for cutoffs at 5, 10, 20, and 30% Nyquist cutoff using a 10% Nyquist transition band, and passband weight $\alpha = 1$. Note that one should not pick a cutoff that is too small a percentage of Nyquist, because it is hard to design a good filter in that case. This means one should not pick a sample time T that is too short. Error in the passband that aims to make $H(z) = 1$, results in a nonzero forcing function in the difference Equation 2.17 making the error not converge to zero even below the cutoff. So design for high accuracy of $H(z)$ in the passband.

There are several subtleties to consider. There are two ways to implement the control law Equation 2.16. One can use $\phi F(z)$ to filter the error, then add it to $U(z)$ using $H(z)$,

and create the filter $\phi H(z)F(z)$ and apply it to the error $E(z)$. In the former case, one is still using $F(z)$ with its possibly large gains and large range of gain magnitudes, and possible numerical loss of accuracy. In that case one may want to introduce $a^T W_a a$ into the cost function. In the later case, these large gains involved in the latter approach.

Another subtlety relates to the cutoff intended and the cutoff achieved. Reference [8] and a reference therein, discusses the difference between the cutoff frequency used in Equation 2.19 and the resulting cutoff frequency of the learning as indicated by the sensitivity transfer function from $Y_D(z) - V(z)$ to the resulting error $E(z)$, $S(z) = [z^p - H(z)]/[z^p - H(z)[1 - \phi G(z)F(z)]]$. If one picks the cutoff to be the highest frequency possible and still satisfy stability in Equation 2.19, for example experimentally since one does not know what is wrong with one's model, then one finds that the effective frequency of cutoff observed in the sensitivity transfer function can be much lower. This happens due to some ill-conditioning in $S(z)$ when one tries to eliminate errors far above the bandwidth of the control system $G(z)$. However, if one is picking the cutoff frequency for one of the many other reasons described above, then it is the effective cutoff in $S(z)$ that is of interest, and one may need to iterate with higher cutoffs in $H(z)$, to produced the desired resulting cutoff in $S(z)$.

Sample Rate T

Our RC control law aims to have zero error at the sample times. For high frequencies this is not necessarily a good cancellation of the error. At Nyquist frequency there are 2 samples per oscillation. Figure 2.17 shows a sinusoid sampled at one quarter Nyquist frequency

or 8 samples per period, and run through a zero order hold. The subtraction of these two signals makes zero error at the sample times, but has large error of the same amplitude as the signal itself between sample times. In practice the zero order hold correction comes through the physical system, which has low pass characteristics, and the error between time steps will not be this extreme. Also shown is the zero order hold approximation of the sinusoid when sampled 5 times as fast, and now the zero order hold approximation starts to look reasonably good. This suggests that trying to cancel periodic disturbances too near Nyquist frequency by getting zero error at the sample times is not very effective. One should have perhaps at least 8 samples per oscillation, and this means one might want to limit corrections to frequencies below one quarter Nyquist. Using a fast sample rate with substantially more samples per period produces a much clos-er approximation of the signal, and can produce a much better cancellation of the periodic disturbance, and reduced error in following the periodic command when intersample error is considered. This applies to all frequencies, including those below a desired cutoff. Of course, fast sampling decreases the effects of aliasing as well. But we have seen that without using a cutoff, the gains grow to very large values as the sample rate goes up, Figure 2.8. So one needs a cutoff. The 3rd order system being used as an example here is a reasonable model of the command to response of each joint of a Robotics Research Corporation robot (Reference [8] and references therein). The bandwidth is 1.4Hz, but the sample rate in 200Hz making a Nyquist frequency of 100Hz very far above the system bandwidth. It is essentially impossible to create a model that is good to 100Hz. The best model we could create might be accurate to at most 20Hz. If no extra poles appear above that frequency without corresponding zeros, the amplitude of a command needed to cancel an error near Nyquist is on the order of

magnitude of 10,000 times the size of the error being corrected. One can use a sample rate that is fast compared to the dynamics in the system, but one must limit the frequency range of the RC law to error frequencies that can reasonably be correct, and also to frequencies for which one is able to create a model with some confidence.A filter cutoff $H(z)$ is needed if one wants a fast sample rate, and the sample rate is then limited by not only real time computation considerations, but the ability to design a cutoff filter that is accurate in the passband. Recall that a sample rate so fast that one wants a cutoff at 5% Nyquist does not allow a reasonable cutoff filter design when using 51 gains, Figure 2.15. We study this combination of a fast sample rate with cutoff compared to a slower sample rate. Figure 2.4 designed a compensator $F(z)$ for the standard $3^r d$ order system with a 200 Hz sample rate, or 100 Hz Nyquist frequency. The largest gain in magnitude in $F(z)$ for this sample rate is $-4,897$. When the sample rate is increased to 1000 Hz (Nyquist frequency of 500 Hz), this gain becomes $-597,721$. Then we design a 51 gains zero phase filter with 20% Nyuist cutoff, i.e. 100 Hz, and the combined filter $H(z)F(z)$ is computed having 62 total gains. The maximum of these gains in magnitude is -1065, which very much reduces concerns about a large range of gain sizes, and still allows a sample rate of 1000 Hz. Figure 2.18 shows the left hand side of Equation 2.19, the approximate decay rate of error at each frequency when using 1000 Hz sample rate and a cutoff filter at 20% Nyquist. We see that it is comparable below the 100 Hz cutoff to the learning rate shown in Figure 2.4 using $F(z)$ designed for 100 Hz Nyquist instead, but now one gets this performance with the fidelity of signal representation associated with 1000 Hz, and with somewhat lower gains, but at the expense of using an increased number of gains.

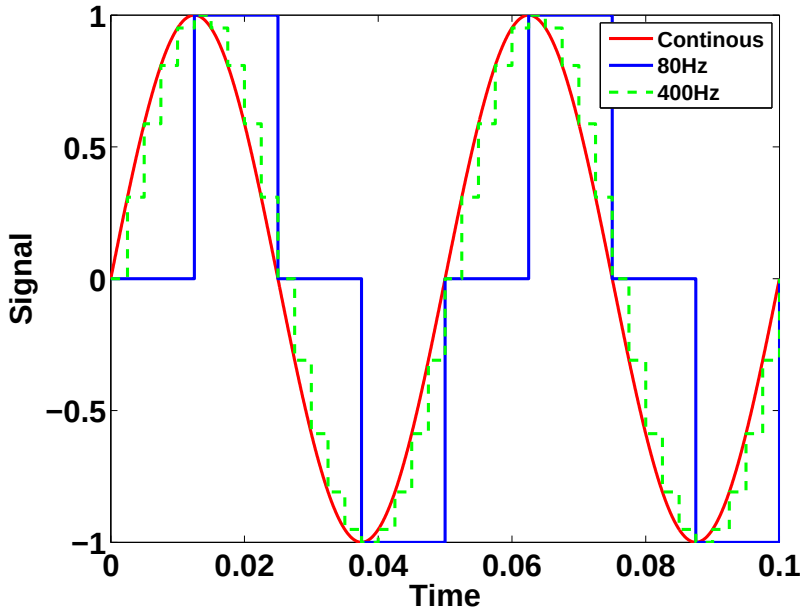

Figure 2.17: Zero order hold approximation of a sinusoid with different sample rate

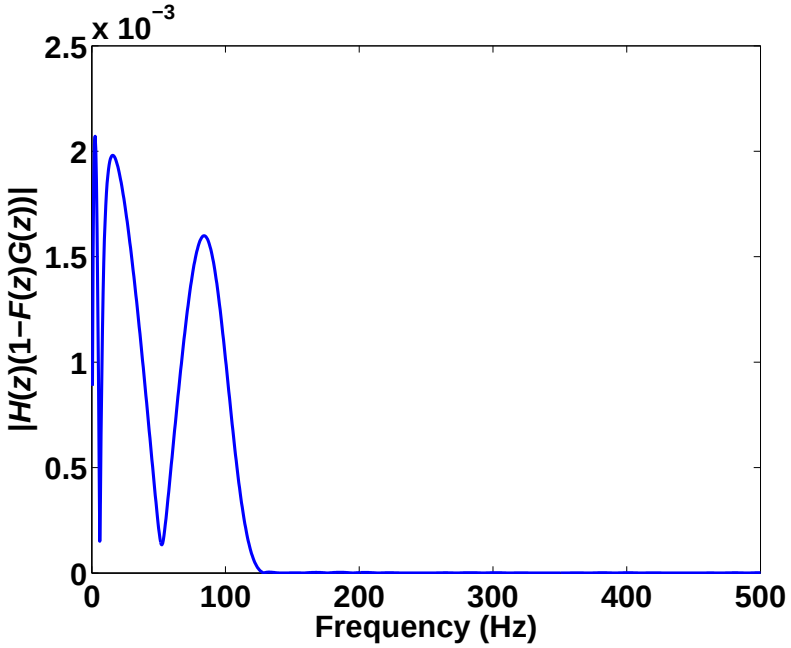

Figure 2.18: The result of using 1000 Hz sample rate with a 100 Hz cutoff filter

Adjusting RC gain ϕ

Now consider adjusting the gain ϕ. Note that if $|1 - F(z)G(z)| < 1$m i.e. if $0 < F(z)G(z) < 2$, then $|1 - \phi F(z)G(z)| < 1$ for $0 < \phi < 1$. So reducing the gain of the RC will not destabilize the system, but it slows the learning from one repetition to the next. It does reduce the size of the gains, but to make a big change results in very slow learning. When $F(z)G(z) \neq 1$ due to model error, a smaller ϕ can reduce gains, reduce amplification of noise in the command signal, and improve robustness, but the RC still asks for zero error for all harmonics up to Nyquist, so that issues related to hardware limits and energy expenditure are not addressed unless one also includes a zero phase filter cutoff. Also, consider an-other performance characteristic that we have not been addressing, the error in the output from disturbances at frequencies other than those addressed of period p time steps. If $F(z)$ is such that $F(z)G(z)$ can be considered unity, then the transfer function using Equation 2.17 from $V(z)$ to error $E(z)$ becomes $[z^p - 1]/[z^p - (1 - \phi)]$. For $\phi = 1$ this equals $1 - z^{-p}$. For a disturbance $V(z)$ at a frequency half way between two harmonics, this doubles the size of the disturbance seen in the output. For reduced ϕ the amplification of these becomes $2/(2 - \phi)$ which reduces to no amplification as the learning speed gets arbitrarily slow.

Picking a reduced upper limit on frequency in cost function

If we know that we will use a filter $H(z)$ with a certain cutoff frequency, then it is reasonable to say there is no reason for the cost function used to fit $F(z)$ to $G^{-1}(z)$ to consider matching frequencies well above the cutoff. There is a potential for the design resulting from limiting

ω_N to create a much improved fit below the cutoff. And this would result in faster learning in the frequency range considered. Figure 2.19 shows the values of $1 - F(z)G(z)$ versus frequency for ω_N cutting off at 40, 50, 70% Nyquist, and compared to no such cutoff. Figure 2.20 is a detailed look at frequencies up to 50% Nyquist. Note that the improvement can be a little erratic, since a 40% cutoff is not better within the 40% range than the 50% cutoff. But it is clear that very substantial reductions in the value of $1 - F(z)G(z)$ within the addressed frequency range are possible, with resulting much faster convergence. Comparing the 50% result in the range up to 50 Hz to the corresponding results in Figure 2.4, we see that there is a reduction in $1 - F(z)G(z)$ by about a factor of 100. We note that numerical experience argues against using too small a percent cutoff. Using 200 Hz sample rate, asking for 12 gains to minimize the cost up to 40% gives the result shown, but limiting the cost ω_N to 30% Nyquist results in a Matlab warning of ill-conditioning. At 400 Hz sample rate, 40% Nyquist results in a warning, but not 50% Nyquist.

2.6 Conclusions

Motivation for RC frequency cutoff Repetitive control nominally asks for zero error of a control system executing a periodic command, or in the presence of a periodic disturbance. This means zero error at the fundamental frequency and all harmonics up to Nyquist frequency. This is easily a high frequency, and one therefore normally needs to cutoff the convergence to zero error above some cutoff frequency. There are two classes of motivation for the cutoff. Ones where the designer can probably know the desired cutoff frequency during the design process, include:

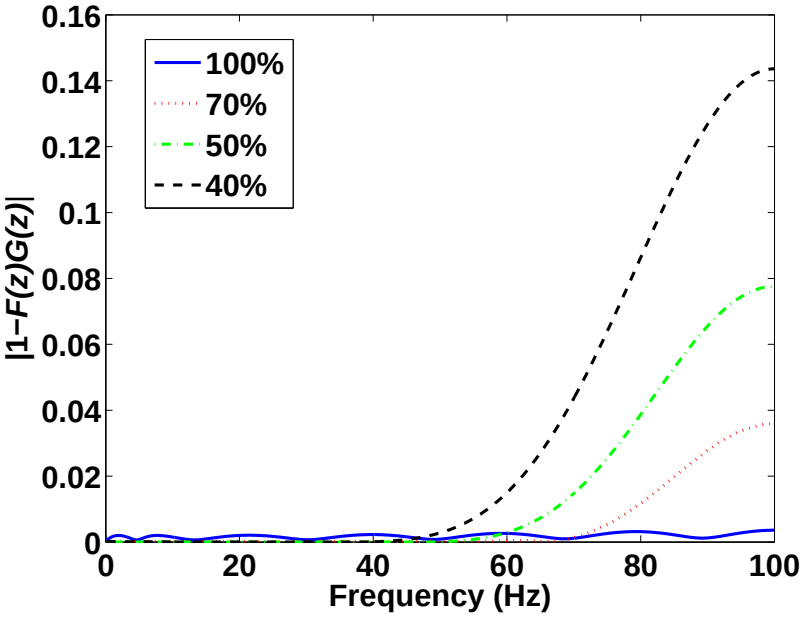

Figure 2.19: $|1 - F(z)G(z)|$ for $F(z)$ designed with several cost function cutoffs ω_N

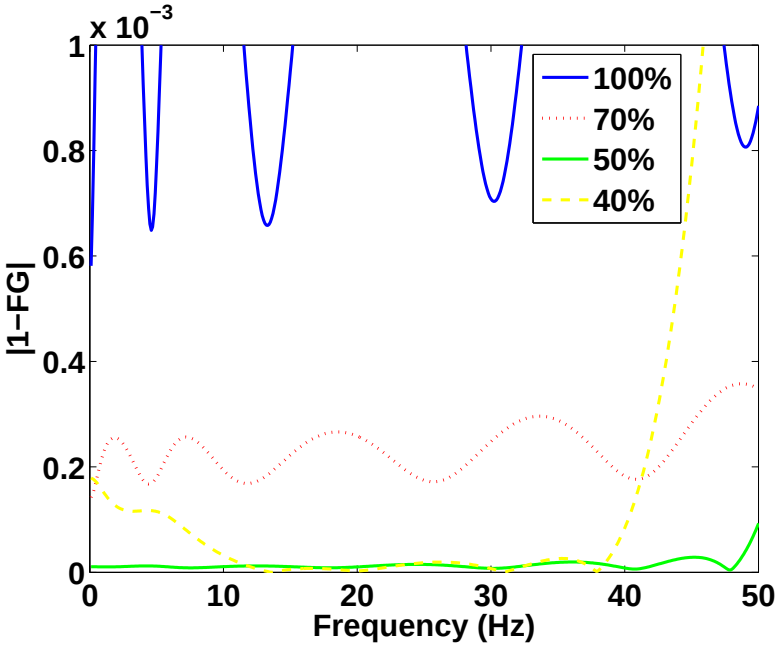

Figure 2.20: Detailed view of Figure 2.19 up to 50% Nyquist

- The frequency content of the error to be corrected can be limited, based on the frequency content of the desired periodic output, and the frequency content of the periodic disturbance.
- The frequency response of the feedback control system whose input is adjusted by the RC system, has its own bandwidth. RC can correct error far above this bandwidth, but eventually correction requires too much control effort.
- Related to this, one may use a frequency cutoff to avoid actuator saturation, or to be kind to the hardware and not work it too hard, or to not use too much energy.
- Perhaps one knows that the system model is not good enough to try to correct above some frequency.

The other class of problems has the objective:

- Correct as much of the error as possible, going to as high a frequency as possible.
- Model error is usually the limiting factor, and when designing the RC system, one does not know what is wrong with the model. Hence, one needs to find the cutoff by tuning it experimentally. Pick a generous cutoff, observe the RC behavior, if error is growing find the frequency components that are growing to determine the needed cutoff.

Types of Cutoffs

- $H(z)$ must be the primary cutoff. It is the only cutoff that applies not only to the error signal$E(z)$, but to the whole accumulated command signal $U(z)$. This is usually

needed for stability robustness to high frequency model error. An imperfect cutoff applied to $E(z)$ cannot supply this robustness.

- ω_N can be used to make $F(z)$ be a more perfect model of $G^{-1}(z)$ in the region below the cutoff frequency. It should be chosen somewhat above the cutoff in $H(z)$ to ensure good performance up through the $H(z)$ cutoff frequency, and of course it should be chosen to not go too far into frequencies where the model is no longer accurate.
- W_a in some ways acts as a cutoff, but should be used for purposes of decreasing gain magnitudes. It can improve robustness to error in some frequency range by slowing the learning rate, but lack of sufficient attention at high frequencies can produce instability and need $H(z)$.

Possible design objectives when the cutoff frequency is known

- One may want to converge as fast as possible. The choice of n and m and ω_N assist here.
- One may want to use a particularly fast sample rate, making T small. This can improve the cancellation of the continuous time periodic disturbance using a discrete time control law. It can also help reduce aliasing. And when the period is not an integer number of time steps it reduces error wither or not interpolation is use. One is limited by real time computation requirements, and one cannot design a good cutoff $H(z)$ if the cutoff is too low a percent of Nyquist frequency.
- One may need to reduce the amplification of noise. RC overall gain ϕ reduces throughout all frequencies, use of W_a targets more the high frequencies. Choice

of ω_N can reduce the size of gains, and so does a reduced cutoff in $H(z)$.

- There may be disturbances that are not periodic with the given period. With $\phi = 1$, the error from these disturbances will be doubled for frequency components half way between two harmonics. Reducing ϕ toward zero will come arbitrarily close to not amplifying these errors, but at the expense of arbitrarily slow learning.

Additional considerations when one wants as high a cutoff as possible

- Tuning the cutoff of $H(z)$ experimentally allows one to go up to that frequency at which the model error involved in designing $F(z)$ is just on the verge of producing instability, the highest cutoff possible. Note that the instability is likely to grow slowly in many applications.
- Reducing ϕ pulls the $\phi F(z)G(z)$ curve toward the origin, or the boundary of stability, but it also gets more of the curve inside the unit circle, and hence, improves robustness and can allow higher cutoff at the expense of slower learning.
- If one has a range of parameter uncertainty in one's model, one can improve robustness to model error by designing an $F(z)$ to minimize a sum of cost functions for models in the distribution, Reference [19]. One can go further, and make ϕ a function of frequency $\Phi(z)$ slowing the learning where one needs more robustness, Reference [18].

Chapter 3

Cross Fertilization Between Iterative Learning Control and Repetitive Control

Existing ILC laws such as transpose law, partial isometry law have good robustness to model error, but general RC laws don't. This chapter investigates how ILC laws can be converted for use in RC to improve robustness. Also robustification by adding control penalty in quadratic cost function is compared to use a frequency cutoff filter.

3.1 Introduction

Iterative learning control (ILC) aims to converge to zero tracking error of a feedback control system repeatedly performing the same task, but adjusting the command to the feedback controller each repetition based on the error observed in the previous repetition. Repetitive

control (RC) is similar, but seeks to converge to zero tracking error of a periodic command as time progresses, or to cancel the influence of a periodic disturbance as time progresses, by observing the error in the previous period. These types of control are very similar, but they have rather different conditions for stability because one wants zero error during the transient phase of each repetition or run, and the other seeks zero error as time steps tend to infinity. Repetitive control has application in spacecraft for active vibration isolation of fine pointing equipment when there is vibration from slight imbalance in CMG's or reaction wheels. ILC has application for repeated scanning maneuvers of fine pointing sensors.

ILC is normally formulated in the time domain using state variable models and Markov parameter models. A set of effective ILC laws has been developed in the literature which up-dates the full input history for the next repetition based on the history of the previous repetition [7]. These laws use a learning matrix to do the update, and these are chosen to produce symmetric update matrices that have good numerical robustness because of the orthogonality of the eigenvectors. Considerable experience has been gained with these laws, and good robustness properties are observed to model errors. Each law has the property that the learning rate gets slow at high frequencies.

The first purpose of this chapter is to present and develop methods by which the design approaches of ILC can be converted to use in the RC problem. The decreased learning rate at high frequencies provided by the ILC laws when converted to RC can be seen to improve robustness to model error. Robustness to model error is fundamental to both ILC and RC because they seek zero tracking error in the real world by iterating with the real world, rather than zero error in one's model of the world. The result can be very much improved final error levels approaching the repeatability level of the hardware.

3.2 General ILC Formulation

Both ILC and RC make use of the error measured in the previous repetition or period to adjust the control action. This section gives the general linear ILC formulation in Reference [7] for a SISO system

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ y(k) &= Cx(k) \end{aligned} \tag{3.1}$$

This could be any system, but most often it represents a closed loop feedback control system, and it is desired to iteratively adjust the command input $u(k)$ in order to converge to zero deterministic error $e_j(k) = y^*(k) - y_j(k)$ tracking the desired trajectory $y^*(k), k = 1, 2, \ldots, p$ as the repetition number j tends to infinity. Each repetition starts from the same initial condition. It is convenient to define vectors giving the whole history of the input, the output and desired output, and the error for a repetition, and denote them by under-bars $\underline{u}_j = [u_j(0) \quad u_j(1) \quad \ldots \quad u_j(p-1)]^T$ and similarly for $\underline{y}_j$, $\underline{y}^*$, $\underline{e}_j$ except that the time arguments start at step one and go to step p. Here we assume that the time delay through the system is one step, which implies that $CB \neq 0$. Simple modifications handle other delay values. Define a difference operator $\delta_j \xi = \xi_j - \xi_{j-1}$ for any quantity ξ. Then a general linear learning control law is given by

$$\underline{u}_{j-1} = \underline{u}_j + L\underline{e}_j \tag{3.2}$$

where L is a p by p matrix of learning gains. The repetition domain system model is obtained by writing the convolution sum solution to Equation 3.1 for each time step and packaging

the result in matrix form

$$\delta_{j+1}\underline{y} = P\delta_{j+1}\underline{u}; \quad \delta_{j+1}\underline{e} = -P\delta_{j+1}\underline{u}$$

$$P = \begin{bmatrix} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & CA^{p-3}B & \dots & CB \end{bmatrix} \tag{3.3}$$

Matrix P is a Toeplitz matrix. Using $\delta_j\underline{y} = -\delta_j\underline{e}$ one can write the error propagation equation

$$\underline{e}_{j-1} = (I - PL)\underline{e}_j \tag{3.4}$$

where I is the identity matrix. Then the condition guaranteeing convergence to zero tracking error as the repetitions go to infinity, and the condition for monotonic delay of the Euclidean norm of the error with repetitions, are

$$\begin{gathered} |\lambda_i(I - PL)| < 1 \quad i = 1, 2, \dots, p \\ \|I - PL\|_2 < 1 \end{gathered} \tag{3.5}$$

respectively, where λ_i represents the i^{th} eigenvalue, and subscript 2 indicates the induced Euclidean norm given by the maximum singular value of the matrix.

Four ILC Control Laws

Four choices for ILC laws are summarized in this section. The first law is the simplest possible ILC, one with a very natural motivation, and it is the initial law developed in the

field. It has almost ideal stability robustness, converging to zero error nearly independent of the system dynamics, but it does not have good learning transients and can only be used effectively on first order systems (Reference [7], [20]). The three remaining laws are very effective ILC laws used in practice.

Simplest Control Law This law implements the following concept, if the output at a given time step in the previous run was two units too small, add two units to the command in this run, at the appropriate time step. Note that we assume a one time-step delay from input to output, so we add two units to the command one step before the error considered. Also, in the spirit of typical classical control, we can insert a scalar learning gain ϕ so that instead of asking for 2 units more, we ask for 2ϕ. Perhaps being a bit less aggressive and asking for less change in the next iteration can have some benefit. Then the learning gain matrix L is a p by p identity matrix multiplied by this scalar learning gain

$$L = \phi I \tag{3.6}$$

P transpose Law The following ILC law is called the contract mapping law or P transpose law (Reference [7],[10]). It is a contraction mapping in the sense of the Euclidean norm of the tracking error from iteration to iteration. The law is given as

$$L = \phi P^T \tag{3.7}$$

Partial Isometry Law P transpose law has the undesirable property of learning high frequency components of the error very slowly in most applications. The partial isometry

ILC law (Reference [7], [10], [21]), helps to address this issue. Let the singular value decomposition of matrix P be given by

$$P = USV^T \tag{3.8}$$

Here U and V are unitary matrices whose columns (and rows) represent unit vectors in p dimensional space, and these vectors are orthogonal. As a result, the inverse of the these matrices is given by their transpose, $U^{-1} = U^T$ and, $V^{-1} = V^T$. Note also that all of the eigenvalues of P in Equation 3.3 are equal to CB which is nonzero. Therefor P is full rank and all the singular values in the diagonal matrix $S = diag(\sigma_1, \sigma_2, \cdots, \sigma_p)$ are nonzero and positive, and it is guaranteed to have an inverse (however, it is usually badly ill-conditioned). The partial isometry law is given by

$$L = \Phi U V^T \tag{3.9}$$

Quadratic Cost Law In reference [12], the learning updates each iteration are tuned by minimizing a quadratic cost function with a parameter r_d that adjusts the speed of learning and r_u governs how far from zero error one will be once the learning process converges.

$$J_j = \underline{e}_j^T Q \underline{e}_j + \delta_j \underline{u}^T \overline{R}_d \delta_j \underline{u} + \underline{u}_j^T \overline{R}_u \underline{u}_j \tag{3.10}$$

The standard version of the quadratic cost law (References [22], [23]) do not have the last term in the cost. This term is sometimes included to avoid effectively inverting the ill-conditioned matrix. Reference [12] uses this term to maintain zero tracking error up to

some frequency, and then impose a small penalty. This forms an alternative to using a cutoff filter. Define hat quantities as follows

$$\hat{e}_j = U^T \underline{e}_j; \hat{u}_j = V^T u_j; \hat{w}_j = U^T \underline{w}_j; \overline{R}_d = V R_d V^T; \overline{R}_u = V R_u V^T \tag{3.11}$$

By converting to the hat variables we convert to what becomes the discrete frequency components as the value of p gets large (Reference [24]). We will denote the k^th component of the hat variables as an argument, where we start with one and progress to p for all variables including $\hat{u}_j$. Then as p gets large, k denotes a frequency component, each frequency having two components corresponding to the need for sine and cosine to span each frequency's space. The cost function can now be written in terms of these new variables as

$$\begin{gathered} J = \hat{e}_j^T \hat{e}_j + \delta_j \hat{u}^T R_d \delta_j \hat{u} + \hat{u}_j^T R \hat{u}_j \\ R_d = diag(r_d(1), r_d(2), \cdots, r_d(p)); R_u = diag(r_u(1), r_u(2), \cdots, r_u(p)); \end{gathered} \tag{3.12}$$

We choose to pick the weight matrices diagonal in which case, as p gets large we are supplying weighting factors for each frequency component. As a result, the cost can be decomposed frequency by frequency, so that one solves the optimization problem independently for each k

$$J_j = \sum_{k=1}^{p} J_j(k); J_j(k) = \hat{e}_j^2(k) + r_d(k)(\delta_j \hat{u}(k))^2 + r_u(k)(\hat{u}_j(k))^2 \tag{3.13}$$

In order to determine the optimal change to make in the command input, $\delta_j \hat{u}(k)$, one computes $dJ_j(k)/d\delta_j \hat{u}(k) = 0$. The dependence of the last term on the change in command

is $\hat{u}_j = \hat{u}_{j-1} + \delta_j \hat{u}(k)$. The dependence in the first term is computed as follows. From $\delta_j \underline{e} = -P\delta_j \underline{u}$ or $\underline{e}_j = \underline{e}_{j-1} - P\delta_j \underline{u}$, then converting to hat variables results in the uncoupled set of equations $\hat{e}_j = \hat{e}_{j-1} - S\delta_j \hat{u}$. The needed relationship has the k^{th} component given as $\hat{e}_j(k) = \hat{e}_{j-1}(k) - \sigma(k)\delta_j \hat{u}(k)$ where $\sigma(k)$ is the k^{th} singular value. The result for k is

$$\begin{gathered} \hat{u}_j(k) = H(k)[\hat{u}_{j-1}(k) + F_1(k)\hat{e}_{j-1}(k)] \\ H(k) = \frac{\sigma^2(k) + r_d(k)}{\sigma^2(k) + r_d(k) + r_u(k)}; F_1(k) = \frac{\sigma(k)}{\sigma^2(k) + r_d(k)} \end{gathered} \tag{3.14}$$

One can return to the original un-hatted variables according to

$$\begin{gathered} \underline{u}_j = H[\underline{u}_{j-1} + V^T diag(1/(\sigma^2(k) + r_d(k)))V^T V S U^T \underline{e}_{j-1}] \\ \underline{u}_j = H[\underline{u}_{j-1} + \underline{F} P^T \underline{e}_{j-1}] \\ H = V^T diag(H(k))V; \underline{F} = V^T diag(1/(\sigma^2(k) + r_d(k)))V^T \end{gathered} \tag{3.15}$$

Unifying the set of ILC laws using the quadratic cost structure

The quadratic cost function in Equation 3.10 contains three symmetric matrices whose entries are to be chosen by the designer. If we wish to converge to zero error, set $r_n = 0$. Then Reference [10] gives a general approach to show that the set of choices is sufficiently rich that one can make a quadratic cost to generate each of the other ILC laws. Use the following quadratic cost function

$$J_{j+1} = \underline{e}_{j+1}^T Q \underline{e}_{j+1} + \delta_{j+1} \underline{u}^T R \delta \underline{u} \tag{3.16}$$

Table 3.1: Producing the ILC Laws from the General Quadratic Law

ILC Law	Weighting Matrices
P Transpose Law	$Q = \phi I$, $R = I - \phi P^T P$
Partial Isometry Law	$Q = \phi I$, $R = P^T U(I - \phi S)V^T$
Quadratic Cost Law	$Q = \phi I$, $R = rI$

with Q and R symmetric positive definite p by p matrices. Use the error equation in Equation 3.3, and minimize the above cost function to obtain

$$\begin{gathered} dJ_{j+1}/d\delta_{j+1}\underline{u} = -2P^T Q\underline{e}_j + 2(P^T QP + R)\delta_{j+1}\underline{u} = 0 \\ \underline{u}_{j+1} = \underline{u}_j + L\underline{e}_j; \quad L = (P^T QP + R)^{-1}P^T Q \end{gathered} \tag{3.17}$$

Table 3.1 gives the different weighting matrices for the various ILC laws.

Reference [25] presents another more sophisticated quadratic cost design with more sophisticated choices for the weights Q and R, given by $Q = I$, $R = Vdiag(\alpha_i^2)V^T$, substituting this into the L of Equation 3.17 produces

$$L = Vdiag(\frac{\sigma_i}{\alpha_i^2 + \sigma_i^2})U^T = V\Sigma U^T \tag{3.18}$$

And the decoupled monotonic decay of each component of the error in the U coordinate system obeys

$$\epsilon_{j+1}(i) = \left(1 - \frac{\sigma_i}{\alpha_i^2 + \sigma_i^2}\right)\epsilon_j(i) = \left(\frac{\alpha_i^2}{\alpha_i^2 + \sigma_i^2}\right)\epsilon_j(i) \tag{3.19}$$

This demonstrates that instead of picking one of the control laws above, one can tailor the learning rate for each component of the error on the unit vector columns of U. If one chooses the partial isometry law with $L = V\Sigma U^T$, $\Sigma = diag(\alpha_i)$, which results in the error propagation equation $\epsilon_{j+1}(i) = (1 - \sigma_i\alpha_i)\epsilon_j(i)$. Then one picks any desired decay rate by

Table 3.2: Summary of Learning Rate

ILC Law	Weighting Matrices
P Transpose Law	$1 - \phi\sigma_i^2$
Partial Isometry Law	$1 - \phi\sigma_i$
Quadratic Cost Law	$1 - \phi\sigma_i^2/(\phi\sigma_i^2 + 1)$

adjusting the α_i. In a later section, we will discuss more on this approach for purpose of increasing robustness. Table 3.2 summarizes the learning rates for the above learning laws when picking $\alpha_i = 1$

3.3 RC Fertilized by ILC

The iterative learning control laws are designed in the time domain; they ask to converge to zero error at every time step of a finite time trajectory. Hence, it aims for zero error during the transients as well as during whatever part of the trajectory might be considered steady state. On the other hand, repetitive control laws ask to converge as the time step number tends to infinity. Hence, using steady state frequency response concepts is an appropriate approach to use in repetitive control. To fully use the ILC control strategy in the RC domain, the frequency response versions of the ILC laws need to be presented. Currently there are two approaches which fulfills our needs. First one builds a connection or mapping between Singular Value Decomposition of P and frequency response, the second one shows us the frequency response versions of ILC laws.

Singular Value Decomposition and Frequency Response

Reference [24] presents the relationship between singular values and singular vectors of the matrix P and the frequency response of the system in Equation 3.1 whose transfer function is denoted by $G(z)$. Its frequency response version is $G(e^{i\omega T}) = M(\omega)e^{i\theta(\omega)}$.

For our p step long signal u_k, the Discrete Fourier Transform (DFT) can be used to find its frequency content

$$\begin{gathered} U(e^{i\omega_0 n}) = \sum_{k=1}^{p-1} u(k)(e^{i\omega_0 n})^{-k} \\ \omega = (2\pi/p)n = \omega_0 n, n = 0, 1, 2, \cdots, p-1 \end{gathered} \tag{3.20}$$

Define $z_0 = e^{i\omega_0}$,$U_n = U(e^{i\omega_0}) = U(z_0^n)$, and then Equation 3.17 can be written in matrix form as

$$\underline{U} = H\underline{u}; \quad H = \begin{bmatrix} (z_0^0)^0 & (z_0^0)^{-1} & \dots & (z_0^0)^{-(p-1)} \\ (z_0^1)^0 & x_{22} & \dots & (z_0^1)^{-(p-1)} \\ \vdots & \vdots & \ddots & \vdots \\ (z_0^{p-1})^0 & (z_0^{p-1})^{-1} & \dots & (z_0^{p-1})^{-(p-1)} \end{bmatrix} \tag{3.21}$$

The Inverse Discrete Fourier Transfer is generated using $H^{-1} = (1/p)(H^\star)^T$, $\underline{u} = H^{-1}\underline{U}$, where superscript asterisk denotes the complex conjugate. Now, assuming zero initial conditions, $\underline{y} = P\underline{u}$. Multiplying on the left by H on both sides, and inserting $H^{-1}H$ in front of $\underline{u}$ produces

$$\underline{Y} = E\underline{U}; \quad E = (1/p)HP(H^\star)^T = \hat{H}P\hat{H^\star}^T \tag{3.22}$$

Then E gives the relationship between the frequency components of the input and the frequency components of the output. The $\hat{H}$ represents H with the complex columns and rows normalized to unit length: $\hat{H} = (1/\sqrt{p})H$, $\hat{H}^{-1} = (\hat{H}^{\star})^T$.

As trajectory gets long, the E of Equation 3.22 converges to the system frequency response given by

$$E = diag(M_0 e^{i\theta_0}, M_1 e^{i\theta_1}, \cdots, M_{p-1} e^{i\theta_{p-1}}) \tag{3.23}$$

Write the singular value decomposition of $P = U\Sigma V^T$. Since all of the attenuation or amplification information is contained in the M_n and Σ, by deleting both, the phase information can be determined as

$$diag(e^{i\theta_0}, e^{i\theta_1}, \cdots, e^{i\theta_{p-1}}) = \hat{H} U V^T \hat{H^{\star}}^T \tag{3.24}$$

And we also can have a mapping between the SVD of P and the system magnitude response. For j going from 0 up to Nyquist in the middle of the matrix $\sigma_j = M_j$ and for the rest of the j up to $p-1$, use $\sigma_j = M_j = M_{p-j}$.

Frequency Response Version of ILC Laws

Now consider the frequency response version of the ILC laws (Reference [12]). Denote the transfer function of the system in Equation 3.1 as

$$G(z) = C(zI - A)^{-1}B \tag{3.25}$$

The product $-P\delta_{j+1}\underline{u}$ in Equation 3.3 represents a convolution sum of the input history

Table 3.3: Frequency Response Version of ILC

ILC Law	Frequency Response Version
P Transpose Law	$\phi G(z^{-1})$
Partial Isometry Law	$\phi G(z^{-1})/\|G(z)\|$
Quadratic Cost Law	$[G(z^{-1})G(z)+r]^{-1}G(z^{-1})$

with the unit pulse response history, in $z-$transform space, the error equation can be written in terms of transforms for infinite sequences as

$$E_{j+1}(z) = E_j(z) - G(z)\delta_{j+1}U(z) \tag{3.26}$$

Note that the Taylor series expansion of the $z-$transfer function, expanded in powers of z^{-1} is

$$G(z) = CBz^{-1} + CABz^{-2} + CA^2Bz^{-3} + \cdots \tag{3.27}$$

Examine the product involving the transpose of P in the P transpose law. It is clear that it involves the convolution product using

$$G(z) = CBz^{1} + CABz^{2} + CA^2Bz^{3} + \cdots \tag{3.28}$$

By substituting $z = e^{i\omega T}$, we can have the frequency response of $G(z)$. Since $G(e^{-i\omega T})$ is the complex conjugate of $G(e^{i\omega T})$, they have the same magnitude response, but with the sign of the phase reversed. From the above, the frequency response versions of the ILC laws can be given as in Table 3.3

The frequency response version of the stability condition Equation 3.5 is the approxi-

mate monotonic decay condition

$$|1 - G(z)L(z)| < \quad \forall z = e^{i\omega T} \tag{3.29}$$

Reference [7] proves that this a repetitive control law converges to zero tracking error for all possible periods p, fi and only if this condition is satisfied.

Implementation with FIR Filter for Repetitive Control

In order to create an implementable repetitive controller base on the above frequency response behaviors, one can design an FIR filer with the coefficients for the filter can be chosen to minimize the cost function

$$\begin{aligned} L_R(z) &= a_1 z^{m-1} + a_2 z^{m-2} + \cdots + a_m z^0 + \cdots + a_{n-1} z^{-(n-m-1)} \\ &= (a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_m z^{n-m} + \cdots + a_{n-1} z^1 + a_n z^0)/z^{(n-m)} \end{aligned} \tag{3.30}$$

$$J_{ILC2RC} = \sum_{j=0}^{N} [1 - L_R(e^{i\omega_j T}) L^{-1}(e^{i\omega_j T})] W_j [1 - L_R(e^{i\omega_j T}) L^{-1}(e^{i\omega_j T})]^\star \tag{3.31}$$

$G(z^{-1})$ can also be obtained using the pole and zero form. Considering the transfer function $G(z)$ as

$$G(z) = \frac{K(z - z_1)(z - z_2)}{(z - p_1)(z - p_2)(z - p_3)} \tag{3.32}$$

Replacing z by z^{-1} producing

$$\begin{aligned} G(z^{-1}) &= \frac{K(\frac{1}{z}-z_1)(\frac{1}{z}-z_2)}{(\frac{1}{z}-p_1)(\frac{1}{z}-p_2)\frac{1}{z}-p_2} \times \frac{z^3}{z^3} \\ &= -\frac{Kzz_1z_2}{p_1p_2p_3}\frac{(z-\frac{1}{z_1})(z-\frac{1}{z_2})}{(z-\frac{1}{p_1})(z-\frac{1}{p_2})(z-\frac{1}{p_3})} \end{aligned} \tag{3.33}$$

Thus, for the P transpose RC law, one needs to have

$$L^{-1}(z^{-1}) = G^{-1}(z^{-1}) = -\left[\frac{p_1p_2p_3}{Kzz_1z_2}\right]\frac{(z-\frac{1}{p_1})(z-\frac{1}{p_2})(z-\frac{1}{p_3})}{(z-\frac{1}{z_1})(z-\frac{1}{z_2})} \tag{3.34}$$

3.4 Evaluation of RC Control Laws

The rate of convergence is of interest as a separate subject, but it is also related to the stability robustness of the RC laws to model errors, a main topic of interest in this chapter. Experience indicates that slower learning can be more robust.

Rate of Convergence to Zero Error for All the RC Laws

From Equation 2.9, one can use approximate quasi-steady state thinking, that the left hand side as the error in the next period which is written as the transfer function in square brackets times the error in the current period. Substituting $z = e^{i\omega T}$ into the transfer function makes it into a frequency transfer function. With the quasi steady state thinking used above, the left hand side of Equation 2.8 is the factor by which the amplitude of the error component at frequency ω decays from one period to the next. Thus, the smaller one makes the left side of the Equation 2.8 over all frequencies up to Nyquist, the faster the control system

converges to zero error.

Now one can compare the learning rate between the RC laws originally developed for RC from Equation 2.15, and also compare the learning rate for RC laws created by transfering ILC laws to RC.

Here we examine the approaches with the system descried in Equation 2.7. Initially, consider that this Laplace transfer function is fed by a zero order hold sampling at 100 Hz, and that the number of time steps in the desired trajectory is $p = 100$.

In Figure 3.1, The plot show that learning rate for all the above RC laws on a $log10$ scale using frequency response from Table 3.3. When implemented one needs to design an FIR filter to mimic the frequency response of the table. The ILC equivalent RC laws (P transpose, Partial Isometry,Quadratic Cost) have much slower late rate at most of the frequency range compared to the original RC law. RC from Equation 2.6 aims to have the same learning rate at all frequencies, while Equation 2.15 will attenuate high frequencies. If one's only objective is faster learning rate, then the original RC law is a better choice.

By using Equation 3.30, 3.31 and 3.34, one can have implementable FIR version of RC laws. Figure 3.2 shows the compensator designed by Equation 2.16 using only 12 gains, indicating that $G(z)L(z)$ is very close to the $+1$ on the real axis, meaning that $L(z)$ is very close the frequency response inverse of $G(z)$. Figure 3.3 and 3.4 show the 12 and 50 gains FIR designs for the P transpose RC compensator with unity DC gain. One can produce similar results for other ILC origin RC laws. Comparing the Figure 3.2,3.3 and 3.4, one observes that many more gains are needed to have an FIR compensator that closely mimics the frequency response dictated in Table 3.3 for RC designs based on ILC.

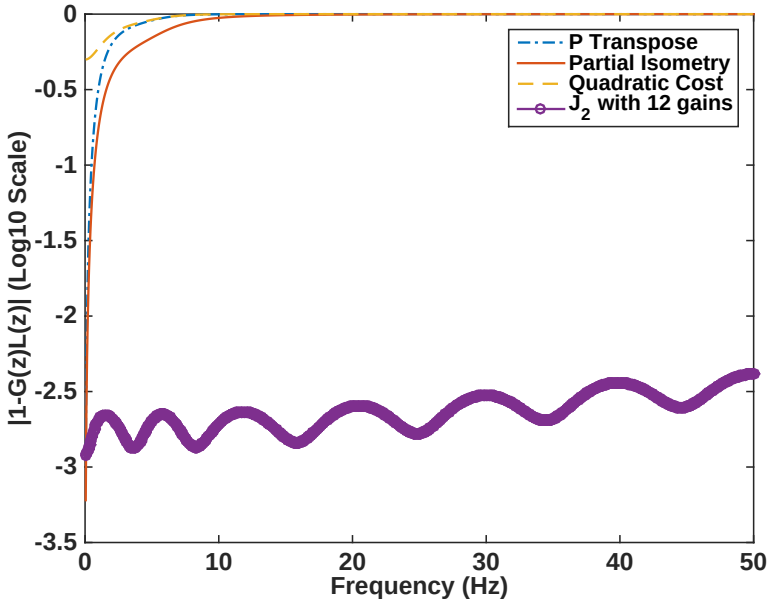

Figure 3.1: Learning Rate of RC laws in Log10 Scale Without FIR Implementation

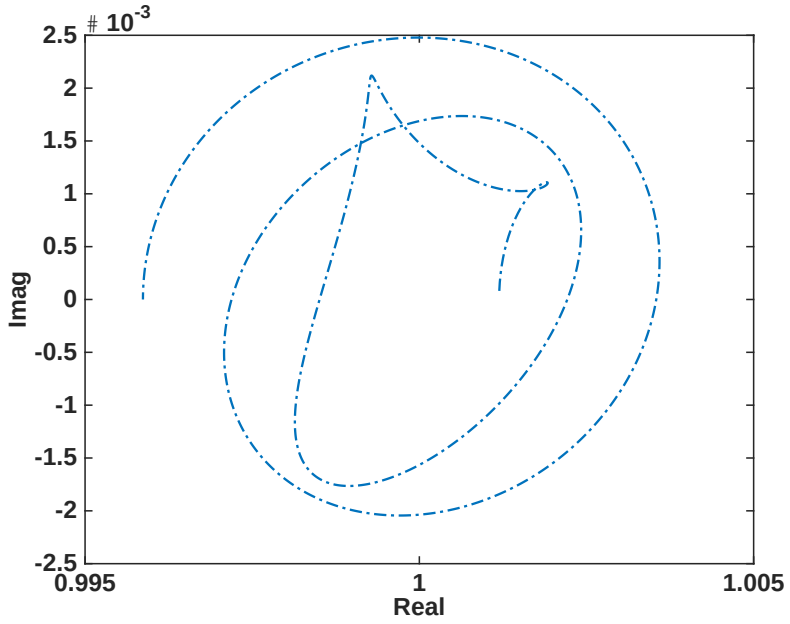

Figure 3.2: $G(e^{i\omega T})L(e^{i\omega T})$ with 12 Gains Design

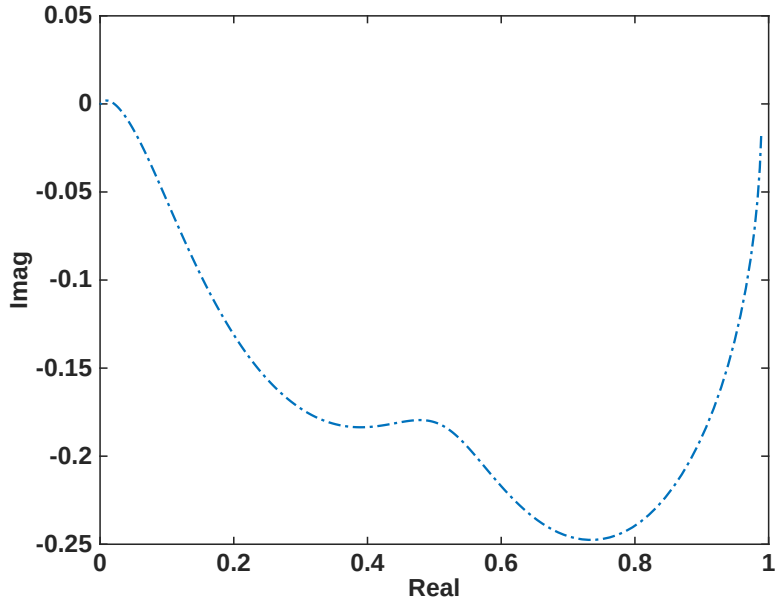

Figure 3.3: $G(e^{i\omega T})L(e^{i\omega T})$with 12 Gains P Transpose FIR Design

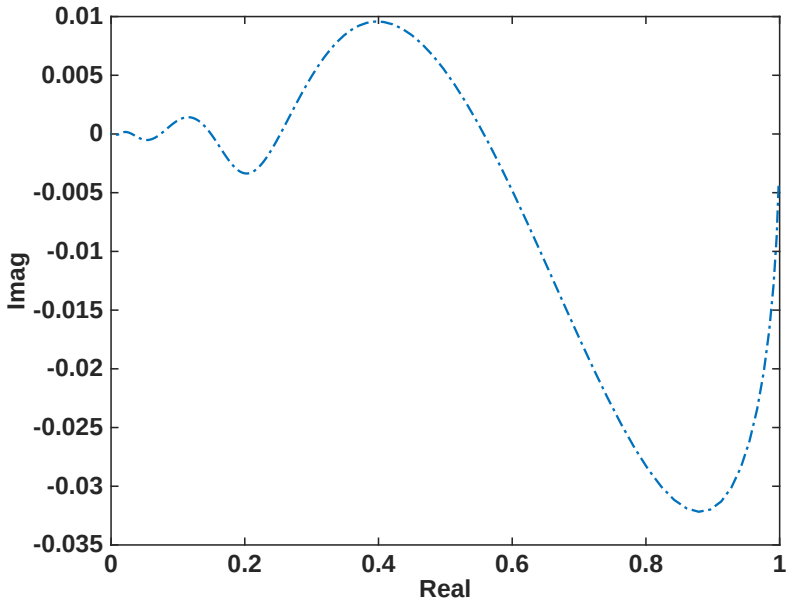

Figure 3.4: $G(e^{i\omega T})L(e^{i\omega T})$ with 50 Gains P Transpose FIR Design

Approach to Evaluate Robustness of Control Laws

Evaluation of stability robustness for each of the learning laws is one of the main interests in this paper. One can define two classes of robustness, robustness to parameter variation, and robustness to unmodeled high frequency dynamics, also called residual modes or parasitic poles. Robustness of the latter kind is addressed by using a zero-phase low-pass filter to cut off the learning, and tuning the cutoff frequency in hardware based on observed performance. In RC it is particularly important to have stability robustness to model errors because RC asks to converge to zero error in the real world, not in our model of the world.

The procedure for studying robustness considered here follows the method used in References [19] and [11]. Consider two sets of uncertainty in each of the parameters in the model Equation 2.7, obeying a uniform distribution ranging $\pm 25\%$ and $\pm 35\%$ around the nominal values respectively, i.e.

$$\begin{gathered} 27.75 < \omega_1 < 46.25; \quad 24.05 < \omega_2 < 49.95 \\ 6.6 < a_1 < 11; \quad 5.72 < a_2 < 11.88; \quad 0.3 < \zeta_1 < 0.5; \quad 0.26 < \zeta_2 < 0.54 \end{gathered} \tag{3.35}$$

Two sets of 1000 samples from each distribution are used to form 2 sets of 1000 models.Comparing the results for each set gives understanding of how much influence the model uncertainty has on the result.The entries in Table 3.4 indicate how many of the models violate the stability condition Equation 3.29 for the column > 1, and how many violate the stability condition by reaching > 1.001 and > 1.01. Because the learning rate can be very slow at high frequencies, there can be many frequencies with magnitude very near 1 on the left of Equation 3.29 and it is hard to know whether a violation of the inequality is due to

Table 3.4: RC Laws Robustness Tests

RC Laws		$\lvert H(z)[1-G(z)L(z)]\rvert$ W/O FIR Implementation			$\lvert H(z)[1-G(z)L(z)]\rvert$ with 12 Gains FIR Implementation		
		>1	>1.001	>1.01	>1	>1.001	>1.01
P Transpose	$\pm 25\%$	0	0	0	0	0	0
	$\pm 35\%$	0	0	0	7	6	3
Partial Isometry	$\pm 25\%$	0	0	0	0	0	0
	$\pm 35\%$	0	0	0	0	0	0
Quadratic Cost	$\pm 25\%$	0	0	0	0	0	0
	$\pm 35\%$	0	0	0	10	9	6
Original RC J Compensator	$\pm 25\%$	-	-	-	25	25	25
	$\pm 35\%$	-	-	-	127	127	123
Original RC with V	$\pm 25\%$	-	-	-	21	21	20
	$\pm 35\%$	-	-	-	114	114	114

round off error in the computation or represents an actual instability. The > 1.001 seems to be a good choice for the threshold, while > 1.01 is normally clearly unstable.

From Table 3.4 it is clear that RC laws created from ILC laws have much better robustness to model parameters uncertainty than the RC designs using costs in Equation 2.15. Adding a small V slows down the learning rate and helps to have better robustness. To further explain the Table, we have Figure 3.5 and 3.6 to give the frequency response plot of the FIR versions P transpose, Partial Isometry and Quadratic Cost Law applied to the nominal system. In Figure 3.5, P transpose and Quadratic Cost have similar curves which means they have similar magnitude and phase error tolerance, and this result is evident in Table 3.4. And Partial Isometry Law had faster learning and as in Figure 3.6, it stays further from the origin than the other two methods, meaning it can still learn when getting close to Nyquist frequency.

Combine the results from Table 3.4, Figure 3.1 and 3.5, one can get better robustness by slowing down the learning rate, and Partial Isometry Law appears to be a good compromise between learning speed and robustness.

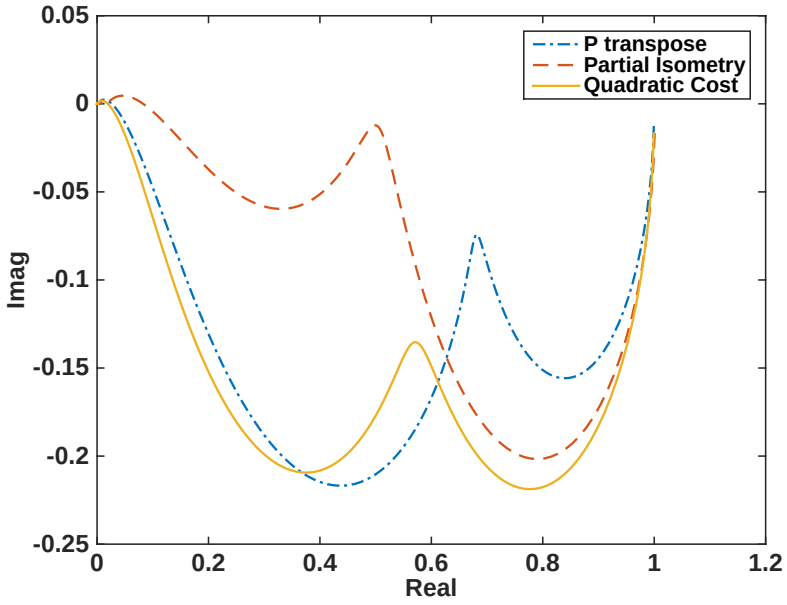

Figure 3.5: $1 - G(e^{i\omega T})L(e^{i\omega T})$ for 12 Gains P Transpose, Partial Isometry and Quadratic Cost FIR Designs

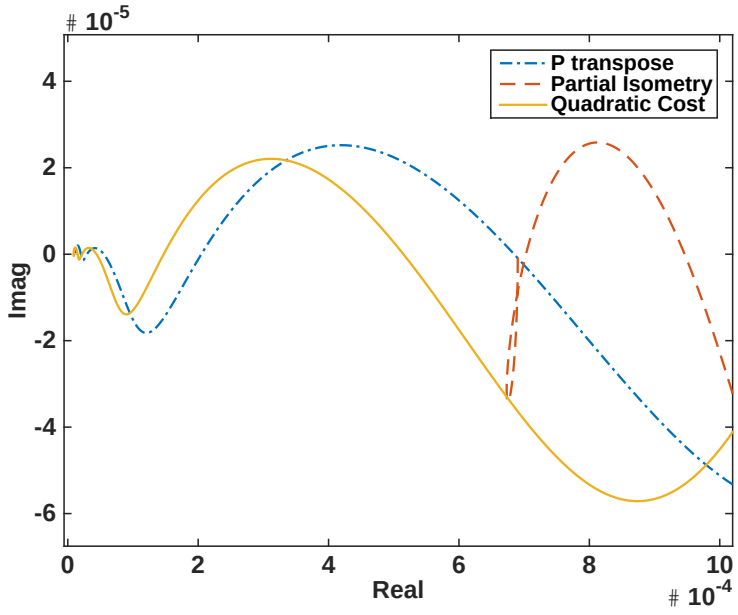

Figure 3.6: Details of Figure 5 near origin

3.5 Handling Robustificaiton of RC to High Frequency Model Error by Penalizing Control Effort

Using the relationship between frequency response and singular values as discussed above, the Equations 3.14 and 3.15 can be converted from ILC to RC

$$\begin{gathered} U_j(e^{i\omega_k T}) = H(e^{i\omega_k T})[U_{j-1}(e^{i\omega_k T}) - F(e^{i\omega_k T})E_{j-1}(e^{i\omega_k T})] \\ H(e^{i\omega_k T}) = \frac{|G(e^{i\omega_k T})|^2 + r_d(k)}{|G(e^{i\omega_k T})|^2 + r_d(k) + r_u(k)}; F(e^{i\omega_k T}) = \frac{G(e^{-i\omega_k T})}{|G(e^{i\omega_k T})|^2 + r_d(k)} \end{gathered} \tag{3.36}$$

Note that the H is a zero phase filter as before, but not a cutoff filter, something that is used to stabilize. One still can use these frequency responses to design an implementable FIR filter and FIR compensator. The design process is to introduce a nonzero value of $r_u(k)$ for only those frequencies where the hardware exhibits growth of error without such a weight on the control action. One stabilizes each such k, but at the expense of nonzero final error at that frequency. The effect of $r_u(k)$ is to expand what was the unit circle stability boundary for this frequency.

In order to have a better understanding of this design, use $E(z) = -G(z)U(z)$ which represents the real world dynamics, and substitute into Equation 2.16. Using the same heuristic argument as for Equation 2.19 gives the stability condition as

$$|H(e^{i\omega_k T})[1 - G(e^{i\omega_k T})F(e^{i\omega_k T})]| < 1 \quad \forall\omega \tag{3.37}$$

In the design process one first deals with adjusting the weights to robustify as needed. For this purpose, we write the stability condition in terms of the original frequency compo-

nents. After making the design and creating the FIR filters, one again checks for satisfaction of Equation 3.37. The condition for picking the weights in the cost function is then

$$\left|1 - \frac{G_M(e^{-i\omega_k T})G_W(e^{i\omega_k T})}{|G_M(e^{i\omega_k T})|^2 + r_d(k)}\right| < \frac{1}{H(e^{i\omega_k T})} = 1 + \frac{r_u(k)}{|G_M(e^{i\omega_k T})| + r_d(k)} \quad \forall k \qquad (3.38)$$

At high frequencies one may not have a good system model, but the amplitude of the output is also very small. This means that it will often be true that the left hand side goes above unity by only a small amount. In this case, and small $r_u(k)$ can be enough to stabilize the system.

To illustrate how r_u works, we introduce a second second order factor with DC gain of unity, an undamped natural frequency $\omega_h = 20$ Hz, and a 0.5 damping ratio. In Figure 3.7, without the zero phase filter, the $|1 - G_W(e^(i\omega_k T))F(ie^(i\omega_k T)|$starts going out of the unit circle after about 20 Hz. By using the stability condition in Equation 3.38, one can a have a zero phase filter as in Figure 3.8 which can help to pull the curve back into unit circle to satisfy the stability condition.

Some Extension The above zero phase filter works very effective for better robustness to high frequency model error. However, the learning rate of quadratic cost RC law is not as fast as other RC laws, and one may ask can the same approach be applied on RC design according to Equation 2.6. The answer is positive.

By using the same stability condition as Equation 3.38 , one can still have a very effective zero phase filter $H(z)$ which helps to overcome the instability caused by high frequency model error, as show in Figure 3.10.

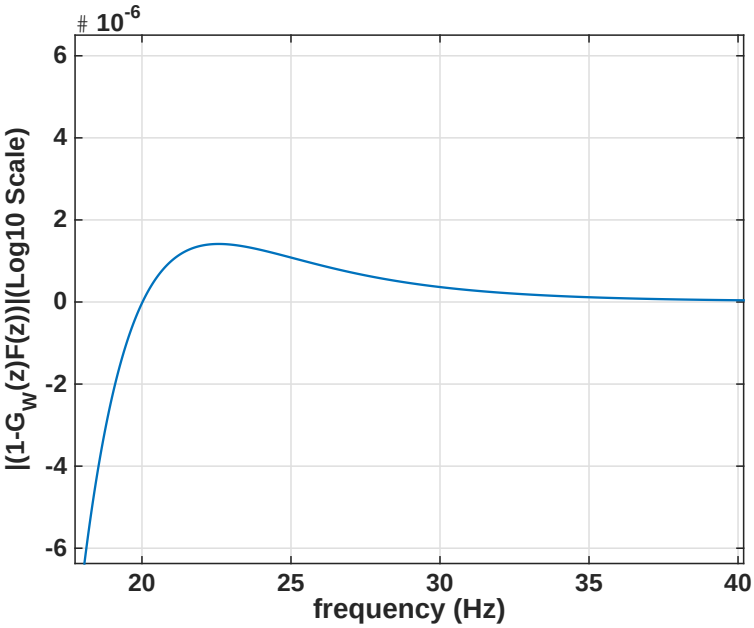

Figure 3.7: $|1 - G(e^{i\omega T})F(e^{i\omega T})|$ without Zero Phase Filter

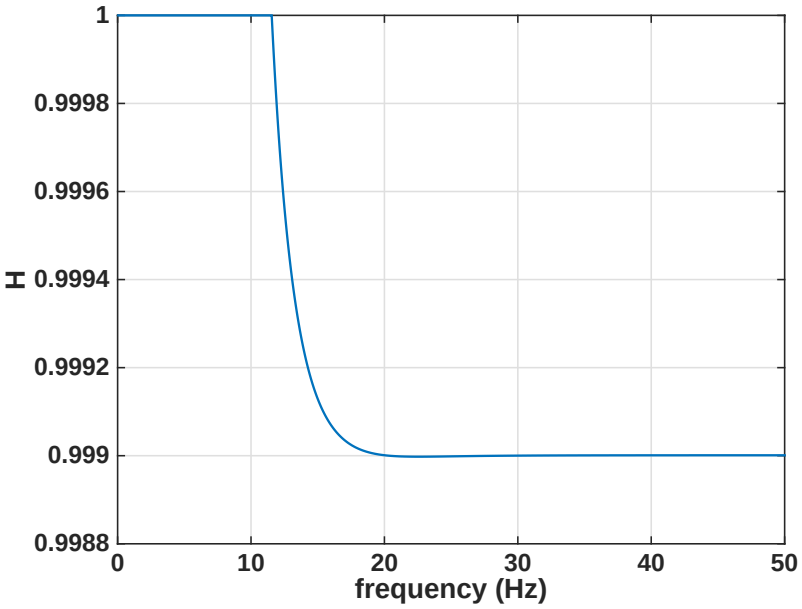

Figure 3.8: Zero Phase Filter H

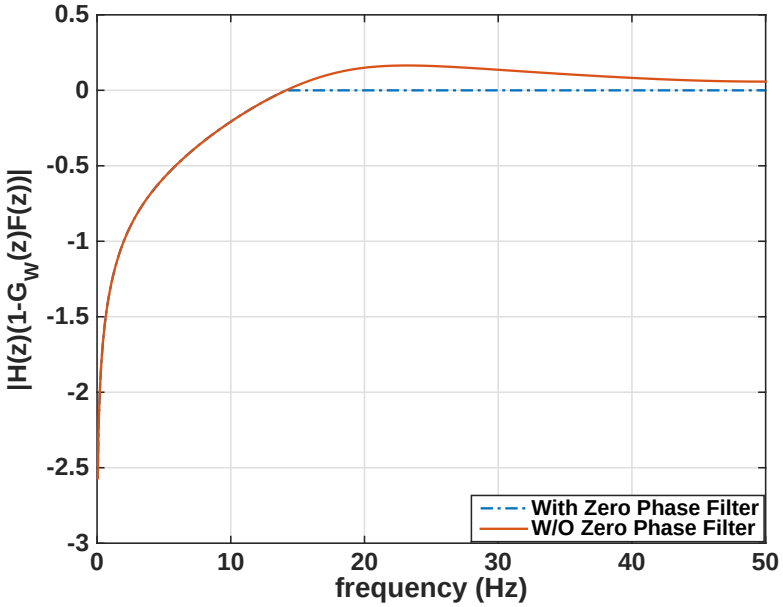

Figure 3.9: $|H(e^{i\omega T})[1 - G(e^{i\omega T})F(e^{i\omega T})]|$ for Effective RC Law by J (in Log20 scale)

Figure 3.10 compares the use of a perfect zero phase low pass filter to the use of control penalty starting at the cutoff frequency. One can evaluate both by computing the sensitivity transfer functions, which tells how much error is produced by allowing a nonzero r_u

$$\begin{aligned} E(z) &= -G_W(z)U(z) + W(z) = S(z)W(z) \\ S(z) &= \frac{z^p - H(z)}{z^p - H(z)[1 - F(z)G_W(z)]} \end{aligned} \tag{3.39}$$

Both approaches help to stabilize the system, but by using r_u one can have more frequencies which partially learning after the cutoff frequency, which can result in a better final error level. One thing we may need to pay extra attention to is that, due to the waterbed effect, the curve of using r_u in Figure 3.10 will go above 1 around 20 Hz to 25 Hz, which means the error in this frequency range will get amplified.

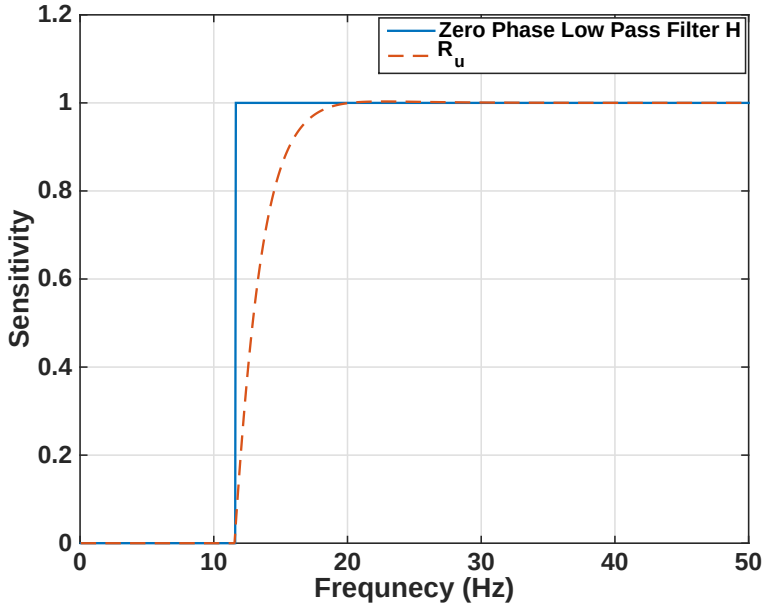

Figure 3.10: Comparison between Zero Phase Low Pass Filter and Weighting Gain R_u

3.6 Conclusions

In this Chapter, both ILC and RC laws are investigated for the purpose of exploring the possibility that ILC laws might be useful in designing effective RC laws. Using effective ILC designs converted to RC has benefits of improving the robustness to model parameter error, which is very important in these fields. Both ILC and RC originally aim to converge to zero error in the real world, not in one's model of the real world, and therefore it is important the laws converge when there is model error. In direct RC design one can always pick the learning rate at each frequency. What the ILC laws do is to suggest an appropriate adjustment of the learning rate to use. Slower learning corresponds to increased robustness. The Partial Isometry Law can be an effective compromise between learning speed and robustness to model error. Averaging of cost functions over model error distributions can improve robustness to parameter uncertainty. One can do the averaging in ILC and then

convert to RC, or convert each model from the distribution to RC and then do averaging. A drawback in ILC designs is slow learning rate at high frequencies.

Chapter 4

Iterative Learning Control Design With Local Learning

Many real-world situations have a linear system with input through a zero-order hold and sampled output. Often one knows the desired output and would like to solve the inverse problem of finding input that produces this output. For the majority of physical systems this results in an unusable input that grows exponentially with time and alternates sign each time step. Recent results demonstrated a new stable inverse method produced by allowing two or more zero order holds between each time step for which one that asks for zero error. This addresses a basic problem and has the potential to address difficulties in many control approaches. In particular, this chapter treats problems such as a factory robot repeatedly starts from a home position, going to a newly arrived object where it performs a high precision task, and then returns to home. High accuracy tracking is only needed for the task part of the trajectory, during which we make use of the stable inverse result.

Other parts of the trajectory use a typical quadratic cost control that compromises tracking accuracy for reduced control effort. The main content of this chapter is to develop a method to create such desired trajectories with a zero-error tracking interval without involving an unstable inverse. Then an easily implementable feedback version is created optimizing the same cost every time step from the current measured position. The above methods are only as good as the model used, so an Iterative Learning Control (ILC) algorithm is created to learn to give local zero error in the real world while using an imperfect model. The approach also gives a method to apply ILC to endpoint problems without specifying an arbitrary trajectory to follow to reach the endpoint. This creates a method for ILC to apply to such problems without asking for accurate tracking of a somewhat arbitrary trajectory to accomplish learning to reach the desired endpoint.

Introduction

There are many situations in engineering where people would like to solve an inverse problem. One application is feedback control systems which are supposed to perform the command they are given, but instead produce a convolution integral of that command. One would like to know the command that causes the feedback system to actually do what you want it to do. This objective can apply to various spacecraft problems requiring high precision motion of scanning sensors. Iterative Learning Control (ILC) is a form of control that learns from one trial or run to the next, aiming to converge to produce the desired output (References [22],[26]]). For a linear continuous time system with Laplace transfer function G(s), the inverse problem wants to compute the needed input from the inverse of the

transfer function. Non-minimum phase systems that have a zero or zeros in the right half plane cause this process to be unstable. Such systems are relatively rare. These systems are said to have intrinsic non-minimum phase zeros. Perhaps the majority of control systems are now implemented using digital control, and this requires that continuous time portions, such as the plant that is fed by a zero order hold, be converted without approximation to a difference equation with the same outputs as the differential equation at the sample times. The resulting discrete time z-transform transfer function G(z) generically has enough zeros introduced (called sampling zeros) that it has one less zero than pole. For pole excesses of 3 or more in continuous time, the 2 or more zeros introduced will have at least one zero outside the unit circle (when the sample rate is not excessively slow), making most discrete time physical systems have unstable discrete time inverses (see Reference [27]). The true inverse in such cases can be useless: it asks for control action that grows exponentially but alternates sign every time step. The error goes through zero at the sample times, and grows exponentially positive then negative between time steps. The zero error at the sample times is hiding the fact that the error between sample times is exploding.

As we mentioned in the previous chapter, ILC observes the stored error samples from the previous time a feedback control system tried to perform the desired trajectory, and based on the error observed in that run, updates the command history to apply in the next run. The aim is to converge to zero tracking error at the sample times. It is not uncommon that the user fails to notice that the iterations will eventually start producing unstable control actions. This can happen because the instability grows slowly. It can look as if convergence has been reached, but with a disappointingly large “converged” error level, while in fact the instability is still “sleeping” and will eventually grow to dominate the response. A series

of publications by the authors and co-workers have developed ILC methods to address this problem (see References [28], [29], [30], [31]).

The difficulty encountered in ILC spawned the sub-problem of finding some kind of stable in- verse. The results are given in Reference [32] with a proof given in Reference [33]. Of course the full inverse for zero error every time step is unique, and it produces unstable time histories of the control action as described above. There must be some modification of the statement of the problem to eliminate the instability. More than one new type of stable inverse is presented in Reference [32] and [33]. The first allows zero tracking error at every time step, with the following exception. If there are m zeros outside the unit circle, one does not ask for zero tracking error in the first m time steps. There are options of what to do for the control in these time steps, but Reference [33] picks the control history of the now underspecified set of equations to minimize the Euclidean norm of the control history. In the case of a 3rd order system with no intrinsic zeros, there will be one sampling zero outside, and for a 5th order there will generally be two. The second stable inverse asks that one increase the sample rate by 2 or 3, ask for zero error every second or third sample time, equivalent to the original sample times. This makes a generalized hold consisting of 2 or 3 zero order holds for each original time step. It is this stable inverse that is used in this paper. These methods are introduced into ILC in References [28], [29], [30] and [31] producing algorithms that converge to zero error at the addressed time steps, with control actions that do not exhibit instability.

The above two kinds of stable inverses are new. There is a recent literature on “stable inverses” that use a different approach, see for example References [34] and [35], and the approach re- quires pre-actuation, extending the desired trajectory to minus infinity in the-

ory. In practice one extends the desired trajectory a sufficient amount of time equivalent to a settling time of zero locations in backward time. Then the tracking accuracy of the desired part of the trajectory is no longer perfect, but can be made as close as desired by making a sufficiently long pre-actuation.

The pre-actuation in the new stable inverse is m time steps equal to the number of zeros outside the unit circle, which is only one time step for a 3^{rd} order pole excess.

The original standard ILC problem asks to converge to zero tracking error at every time step of a finite time desired trajectory. There are classes of applications where this objective is not to- tally appropriate as an overall goal. Instead, only some sub-portions of the desired trajectory need to be performed with high precision. Two important examples: (1) A ubiquitous example is when people use ILC for tracking a trajectory, when the real objective is high precision endpoint control in some point-to-point maneuvers of robots. Learning to create high precision tracking is only important as the robot nears the end point, and the rest of the trajectory can follow any reasonable curve. (2) Another class of problems needs fine accuracy trajectory tracking only during a sub- portion of the trajectory. Consider robots on an assembly line. Often they sit in a home position until some object arrives on a belt, and then moves to the object and performs some task, and then returns to home. The path followed from home to where the task is to be performed, and the re- turn to home, do not need high accuracy tracking, but high accuracy can easily be needed while performing the task, in the middle of the overall trajectory.

Hence, there is a need for ILC algorithms that allow local learning. Reference [36] approaches the problem in an unrelated way using neural networks. Reference [37] offers a method based on wavelet basis functions that could address the problem. References [38]

and [16] address a somewhat generalized version of this problem from the current literature concept of "stable inverse" point of view, using the terminology, output tracking transition switching.

This purpose of this paper is to do the following:

- Develop a method of designing the desired trajectory based on minimizing a typical quadratic tracking cost function that normally represents a compromise between tracking accuracy and control effort. However, this time we double the sample rate for 3^{rd} order pole excess, and introduce the stabilized inverse. This allows us, if we desire, to ask for zero tracking error at every other time step without introducing exponentially growing control action. Then we can adjust the con- trol penalty in the cost function to perform the usual compromise of a quadratic cost through most of the trajectory, but the control penalty can reduce to zero for any part of the trajectory that needs to be followed precisely. It is the new stable inverse result in Reference [32] and [33] that makes this possible. This embeds the zero error high precision portion of the tracking problem into the standard quadratic cost tracking problem in a natural way. The result produces the desired zero error tracking provided the model used is correct.
- Having developed an optimized desired trajectory that one wishes to perform, one might want to implement an optimal feedback to improve performance if disturbances cause some deviation from this trajectory. One can embed the problem of the above item into each time step to create an optimal solution starting from the current disturbed state.

- From the desired trajectory in first item, we develop the corresponding iterative learning control problem whose purpose is to achieve the desired tracking accuracy in spite of some error in the model used in the design. ILC can apply this command history to the physical world and observe the error. A major characteristic of ILC is that it is observing the error in the real world, not in one's model of the real world. ILC laws attempt to be sufficiently robust that given a model of the world used to design the ILC algorithm, the algorithm converges to zero error in the real world model instead of the model, because it is using real world data for the update each iteration. Toward this objective for our given desired trajectory, we incorporate the quadratic cost ILC approach to iterative learning control (References [10], [22] and [23]). This approach has well behaved learning transients and convergence properties. Starting from the quadratic cost based desired trajectory described above, a quadratic penalty on the change in the control from run to run is introduced to create an ILC law that can learn to give zero tracking error in the world when there is some model error in the stable inverse created from a model.

4.1 The Inverse Problem for a State Variable Model

Consider the same discrete time SISO state space model of the Equation 3.1, for a one step time delay through the system, the p step input history vector $\underline{u} = [u(0)u(1)\cdots u(p)]^T$ produces the p step output vector $\underline{y} = [y(0)y(1)\cdots y(p)]^T$. The output for all time steps in this vector can be written as $\underline{y} = W\underline{u} + \overline{A}x(0)$ where $\overline{A}$ is an observability matrix, and W

is the Toeplitz matrix

$$W = \begin{bmatrix} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & CA^{p-3}B & \dots & CB \end{bmatrix} \tag{4.1}$$

Then the unique solution to the inverse problem giving the input needed to produce a given output $\underline{y}$ is given as

$$\underline{u} = W^{-1}[\underline{y} - \overline{A}x(0)] \tag{4.2}$$

Note that the matrix W relates the input and the output histories when the initial conditions are zero, $\underline{y} = W\underline{u}$.

Reference [29], [30] and [31] demonstrate that the matrix W exhibits the instability of the inverse solutions in its singular value decomposition. For every zero outside the unit circle there will be a small singular value that decreases by the singular value to the p^{th} power as the trajectory length p increases. The associated input singular vector components alternate in sign with the time steps (for sampling zeros that lie on the negative real axis) and increase in magnitude each step by the magnitude of the zero location outside the unit circle. The associated output singular vector performs in the same way but with the opposite slope. Thus, to obtain an output that is the output singular vector, an input is required that grows exponentially with time step.

The results on stable inverse summarized in Reference [32] say that if one deletes the number of initial rows of W equal to the number of zeros outside the unit circle, then the

associated singular values are gone, and the singular vector pair exhibiting the instability is gone. Consider for simplicity the situation of a third order pole excess producing one zero outside the unit circle. The more important result for the present objective for this system is: (i) If one eliminates every odd numbered row of matrix W, then it has a stable inverse, or more precisely its minimum Euclidean norm of the control history among the multiple solutions, produces a stable inverse, i.e. the control action produced results in the desired output at every even numbered time step. The equations are underspecified so there are an infinite number of solutions. If there were 2 zeros outside, then eliminate 2 rows between each row that is kept. (ii) An improved version given in References [32] creates the matrix W as a product of a matrix that deals with the zeros outside the unit circle, multiplying a matrix that handles all zeros and poles inside the unit circle. The latter matrix of course has a stable inverse. Then the odd numbered rows of the initial matrix are deleted. Reference [33] proves that this produces a stable inverse, whether or not the zero outside is the image of an intrinsic zero, or it is a sampling zero. For simplicity we use the case of one zero outside to present the method and the results. The methods generalize in an obvious way to more zeros outside.

The results in Reference [33] factorize W as $W = P_O P_I$ where the first matrix deals with the zero outside, and the second deals with all poles and zeros inside. Then the odd rows of P_O are deleted to make a $W_{DOI} = P_{OD} P_I$. Then the solution to $\underline{y}_D = W_{COI}\underline{u}$ used in Reference [33] is given by $\underline{u} = P_I^{-1}(P_{OD})^{+}\underline{y}_D$, and proven to give a stable control action. Superscript $+$ indicates Moore-Penrose pseudo inverse of the underspecified matrix. For

simplicity in this work, we denote W with odd rows deleted by W_D , and use

$$\underline{u} = (W_D)^{+}\underline{y} \tag{4.3}$$

giving that solution that minimizes the Euclidean norm of the resulting $\underline{u}$, which is also observed to give a stable inverse.

4.2 The Model Predictive Control System Model

The problems dealt with in this paper want to use the system model as found in Linear Model Predictive Control (LMPC). For simplicity, the method of generating such a model is presented here starting from a third order scalar difference equation

$$\begin{aligned} \alpha_0 y(k+1) = &-\alpha_1 y(k) - \alpha_2 y(k-1) - \alpha_3 y(k-2) \\ &+\beta_1 u(k) + \beta_2 u(k-1) + \beta_3 u(k-2) \end{aligned} \tag{4.4}$$

Note that the initial conditions required to produce the output at time step 1, are $y(0), y(-1), y(-2)$ and $u(-1), u(-2)$. Then the choice of the initial control $u(0)$ determines the first output y(1) . The LMPC model writes this equation for a chosen number of time steps both forward and backward from any current step of interest k. Considering only 3 steps backward and 3 steps forward, the result is can be packaged as

$$\begin{bmatrix} \alpha_0 & 0 & 0 \\ \alpha_1 & \alpha_0 & 0 \\ \alpha_2 & \alpha_1 & \alpha_0 \end{bmatrix} \begin{bmatrix} y(k+1) \\ y(k+2) \\ y(k+3) \end{bmatrix} = \begin{bmatrix} 0 & \beta_3 & \beta_2 \\ 0 & 0 & \beta_3 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u(k-3) \\ u(k-2) \\ u(k-1) \end{bmatrix}$$
$$- \begin{bmatrix} \alpha_3 & \alpha_2 & \alpha_1 \\ 0 & \alpha_3 & \alpha_2 \\ 0 & 0 & \alpha_3 \end{bmatrix} \begin{bmatrix} y(k-2) \\ y(k-1) \\ y(k) \end{bmatrix} + \begin{bmatrix} \beta_1 & 0 & 0 \\ \beta_2 & \beta_1 & 0 \\ \beta_3 & \beta_2 & \beta_1 \end{bmatrix} \begin{bmatrix} u(k) \\ u(k+1) \\ u(k+2) \end{bmatrix} \tag{4.5}$$

The column vectors in this equation will be denoted by $\underline{\hat{y}}_F(k)$, $\underline{u}_P(k)$, $\underline{\hat{y}}_P(k)$, $\underline{u}_F(k)$ with subscripts indicating future and past. By taking the inverse of the initial matrix and multiplying both sides from the left with it, the model takes the form

$$\underline{\hat{y}}_F(k) = \hat{P}_1 \underline{u}_P(k) - \hat{P}_2 \underline{\hat{y}}_P(k) + \hat{W} \underline{u}_F(k) \tag{4.6}$$

Our initial use of this model is to plan the entire desired trajectory. In this case we set the time step k to zero, and for simplicity of notation we will not show the argument in each of the column vectors. The k argument will be reintroduced when we discuss the possible use of feedback in time when implementing the optimized trajectory.

Planning the whole trajectory dictates the dimensions of the column vectors. The number of entries in the past vectors must be at least as large as the minimum number for setting up the initial conditions, in this case 3. And instead of the 3 used above for the number of entries in the future vectors, they should contain every time step to the end of the problem at time step p, preferably an even number.

Note that if the initial conditions are set to zero, then the two past vectors for $k = 0$

are zero, and the model becomes $\underline{\hat{y}}_F(k) = \hat{W}\underline{u}_F(k)$, demonstrating that the $\hat{W}$ here is the same matrix as the W in the previous section. The model of interest for design of the desired trajectory and for the ILC law, needs to isolate the even numbered time steps in this equation. This requires eliminating the odd numbered entries in column vector $\underline{\hat{y}}_F(0)$ and the odd numbered rows of matrices $\hat{P}_1$,$\hat{P}_2$, and $\hat{W}$. After deletion, we indicate each of the resulting matrices by removing the hat on top of the symbol, except in the case of $\hat{W}$ where we choose to explicitly show the deletion by using the W_D symbol as before. The resulting model after appropriate deletions is

$$\underline{y}_F = P_1\underline{u}_p - P_2\underline{\hat{y}}_p + W_D\underline{u}_F \tag{4.7}$$

With these notation changes, the final equation looks deceptively routine, but it is non-standard because the coefficient matrices are rectangular, and the future vector is limited to even time steps, and the past vector is not.

4.3 A Quadratic Cost Function Based Desired Trajectory With Local Perfect Tracking

Consider a quadratic cost function written for tracking a finite time initial desired trajectory, which we specify at even numbered time steps as $\underline{y}_{FdesA}$

$$J = [\underline{y}_F - \underline{y}_{FdesA}]^T Q[\underline{y}_F - \underline{y}_{FdesA}] + [\underline{u}_F]^T R[\underline{u}_F] \tag{4.8}$$

We want $\underline{y}_F$ to actually equal $\underline{y}_{FdesA}$ for certain critical time intervals. In this work, we consider that we are always far enough away from any actuator constraint limits that they are not active.We specify the rest of the trajectory in some reasonable way. Consider the problem mentioned before of a robot that when something arrives at its station, the robot moves out from a home position to the object where it starts to do its job. After finishing the task, it moves back to the home position to wait for the next item to arrive. While performing the task one wants high precision motion along a prescribed trajectory. But the path from home position to the start of the task can be anything reasonable. Here we ask the user to specify a reasonable path in the associated part of $\underline{y}_{FdesA}$, but since there is nothing special about this choice, we will let the cost function of the above equation create a path that roughly follows the path, doing so in a way that is a trade off between accurately following, and the amount of control effort expended, as penalized by the second term. The resulting trajectory is denoted by $\underline{y}_{FdesB}$, and it is this trajectory we wish to implement in hardware.

Our objective of have accurate tracking for some time steps might suggest that one should adjust the value of Q to emphasize the penalty on the tracking error at those time steps. This only produces perfect tracking asymptotically as the gain is increased to infinity. Instead we set Q to the identity matrix. Then, if the control penalty is zero for some time steps the control action can use whatever control makes the first term zero. Hence, matrix R can be diagonal matrix, and we will adjust the elements differently for different parts of the trajectory.

The stable inverse results we wish to incorporate, require that we restrict the possible values of the control history $\underline{u}_F$ to ones that minimize the Euclidean norm of the possible

control actions producing $\underline{y}_F$ from $\underline{u}_F$ in the underspecified set of equations. To do this, write the singular value decomposition (SVD) of matrix W_D as

$$W_D = U[S \quad 0]V^T = U[S \quad 0]\begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = USV_1^T \tag{4.9}$$

Define a transformed version of the control action as

$$\underline{\mu}_F = V^T\underline{u}_F \qquad \begin{bmatrix} \underline{\mu}_{F1} \\ \underline{\mu}_{F2} \end{bmatrix} = \begin{bmatrix} V_1^T\underline{u}_F \\ V_2^T\underline{u}_F \end{bmatrix} \tag{4.10}$$

The components of $\underline{u}_F$ on V_2, i.e. $\underline{\mu}_{F2}$, have no influence on the output $\underline{y}$ in equation $\underline{y}_F = W_D\underline{u}_F$. These components are orthogonal to the components on V_1, so the minimum norm control action $\underline{u}_F$ that satisfies the equation $\underline{y}_F = W_D\underline{u}_F$ is given by

$$\underline{u}_F = V\underline{\mu}_F = [V_1 \quad V_2]\begin{bmatrix} \underline{\mu}_{F1} \\ \underline{\mu}_{F2} \end{bmatrix} = [V_1 \quad V_2]\begin{bmatrix} \underline{\mu}_{F1} \\ 0 \end{bmatrix} = V_1\underline{\mu}_{F1} \tag{4.11}$$

and we enforce the choice of $\underline{\mu}_{F2} = 0$. Consider the penalty term $\underline{u}_F^T R\underline{u}_F$. The components of $\underline{u}_F$ are for time steps. And we know that for time steps in the part of the trajectory that requires fine tracking we want to eliminate any penalty so that the control action for that part of the trajectory is free to do whatever is needed to make perfect tracking error of $\underline{y}_{FdesA}$ for those time steps. Hence, we want to specify the weight matrix in diagonal form

$$R = diag(r_1, r_2, \cdots, r_p) \tag{4.12}$$

But we must also limit the $\underline{u}_F$ so that the control penalty takes the form

$$\underline{u}_F^T R \underline{u}_F = \underline{\mu}_{F1}^T V_1^T R V_1 \underline{\mu}_{F1} = \underline{\mu}_{F1}^T R_\mu \underline{\mu}_{F1} \tag{4.13}$$

The cost function that results from these modifications and the LMPC model become

$$\begin{aligned} J &= [\underline{y}_F - \underline{y}_{FdesA}]^T [\underline{y}_F - \underline{y}_{FdesA}] + \underline{\mu}_{F1}^T R \underline{\mu}_{F1} \\ R_\mu &= V_1^T R V_1 = V_1 diag(r_1, r_2, \cdots, r_p) V_1 \\ \underline{y}_F &= P_1 \underline{u}_p - P_2 \underline{\hat{y}}_P + US \underline{\mu}_{F1} \end{aligned} \tag{4.14}$$

Substituting the last equation into the cost, differentiating the result with respect to $\hat{\mu}_{F1}$, and solving for this control vector results in the optimal control action

$$\underline{\mu}_{F1} = [R_u + S^2]^{-1} S U^T [\underline{y}_{FdesA} - (P_1 \underline{u}_P - P_2 \underline{\hat{y}_P})] \tag{4.15}$$

This optimal control action $\underline{\mu}_F$ when substituted into the model given by the last equation of Equation 4.14 produces the new optimized desired trajectory $\underline{y}_F = \underline{y}_{FdesB}$

$$\underline{y}_{FdesB} = [US(R_\mu + S^2)^{-1} S U^T] \underline{y}_{FdesA} + I - [US(R_u + S^2)^{-1} S U^T][P_1 \underline{u}_p - P_2 \underline{\hat{y}}_p] \tag{4.16}$$

Comments:

- Note that if the penalty on the control effort R_μ is set to zero, then the square bracket term in the above equation becomes the identity matrix. The initial condition terms disappear from the equation, and $y_{FdesB} = y_{FdesA}$. Then the minimization of the

cost produces the stable inverse solution of Reference [32] giving exact tracking for every other time step of the given desired trajectory.

- Consider a problem such as the robot problem described earlier, that has a section or interval of time steps in the middle of the trajectory for which one wants perfect tracking of y_{FdesA} . In this interval one sets the corresponding entries in diagonal matrix R to zero.
- During the initial part of the trajectory one is not concerned with accuracy. For these time steps one adjust the entries in R as one would do when using quadratic cost control design in the time domain.
- As the time steps approach the high accuracy tracking time interval, one must create a transition zone where the R components of above item are being smoothly transitioned to zero approaching the high accuracy interval.
- After the high accuracy zone one can use these two zones in reverse order.
- An important application is for systems where the really important objective is to reach the desired endpoint with high accuracy. The high accuracy tracking interval could be a chosen inter- val at the end of the trajectory, or it could be just a point. As before, a transition region is needed as the system approaches the high accuracy tracking portion or endpoint.

4.4 Creating Real Time Optimal Feedback

In order to correct for disturbances encountered, one normally wants to have feedback control. Ideally, this feedback gives you the optimal solution from the current known state at

any time step k until the end of the problem. Starting from Equation 4.6 where there was an argument k, we can perform the conversions that resulted in Equation 4.7, but this time keep the argument k

$$\underline{y}_k = P_1\underline{u}_P(k) - P_2\underline{\hat{y}}_P(k) + W_D\underline{u}_F(k) \tag{4.17}$$

Pick k so that the first entry in $\underline{y}_F(k)$ is an even time step. Then the optimal control future input satisfies

$$\underline{u}_F(k) = V_1(k)\underline{\mu}_{F1}(k) = V_1(k)[R_\mu(k)+S^2(k)]^{-1}S(k)U(k)^T\underline{y}_{FdesA}(k) - [P_1\underline{u}_P(k) - P_2\underline{\hat{y}}_P(k)] \tag{4.18}$$

The k dependence is a result of the shrinking number of time steps from the current step to the end. This equation has the form

$$\underline{u}_F(k) = A_1(k)\underline{y}_{FdesA}(k) - A_2(k)\underline{u}_P(k) + A_3(k)\underline{\hat{y}}_P(k) \tag{4.19}$$

The $\underline{y}_P(k)$ and $\underline{u}_P(k)$ are feedback from data, and note that data from both even and odd time steps are requested. The coefficient matrices $A_2(k)$,$A_3(k)$ can be pre-computed for each time step. And one can also do this to the term $A_1(k)\underline{y}_{FdesA}(k)$. The $\underline{u}_F(K)$ of course can be a tall vector with many entries, but the only entries one needs at time step k are the first two entries. Hence one can pre-compute everything needed for the next two time steps from pre-computed gains and the $\underline{y}_P$ and$\underline{u}_P$ data. The pre-computation needed might be substantial, but the on-line computation is minimal and performed every two steps.

4.5 Creating Iterative Learning Control With Local Learning

The method develop to create an optimal $\underline{y}_{FdesB}$ with zero tracking error for the high accuracy portion of the trajectory, produces zero error provided one's model used in the computation is the same as the real world model. A major objective of Iterative Learning Control is to achieve in iterations with the real world, zero error in the world, but the ILC law was designed using an imperfect model. The ILC laws are designed to exhibit convergence robustness for a broad class of model errors around the model used to create the algorithm. Here, for each iteration or run j we optimizes a cost function

$$J_j = [\underline{y}_{F,j} - \underline{y}_{FdesA}]^T[\underline{y}_{F,j} - \underline{y}_{FdesA}] + \underline{\mu}_{F1,j}^T V_1^T R V_1 \underline{\mu}_{F1,j} + \delta_j \underline{\mu}_{F1}^T V_1^T R_L V_1 \delta_j \underline{\mu}_{F1}$$
$$R_L = diag(r_{L1}, r_{L2}, \cdots, r_{Lp}) \tag{4.20}$$
$$\delta_j \underline{\mu}_{F1} = \underline{\mu}_{F1,j} - \underline{\mu}_{F1,j-1}$$

A new penalty term has been added that penalizes the change in the control action from one run to the next. Consider what happens when R is set to zero. This makes a quadratic cost ILC design (References [10], [22] and [23]). Reference [39] shows that this ILC law has good stability robustness properties to model error. The penalty on the change in control action from run to run is purely controlling the learning transients. Convergence is still to zero tracking error for the whole trajectory, because the iterations can accumulate whatever change in control action is needed to get zero error. Note that robustness to model error becomes substantially larger when the learning is made slowly using a large R_L. The learning

law updating the input $\underline{\mu}_{F1,j-1}$ to $\underline{\mu}_{F1,j}$ from iteration $j-1$ to j is

$$\underline{\mu}_{F1,j} = [S^2 + V_1^T(R+R_L)V_1]^{-1}\{V_1^T R_L V_1 \underline{\mu}_{F1,j-1} + SU^T[\underline{y}_{FdesA} - (P_1\underline{u}_P - P_2\hat{\underline{y}}_P)]\} \quad (4.21)$$

To use this ILC law for local learning:

- Make the entries in R become zero for the high accuracy part of the desired trajectory
- During these same time steps, pick R_L entries to correspond to the speed of learning, and the associated robustness to model errors desired.
- The entries in R should transition smoothly to zero as before from the quadratic cost objective to the perfect tracking objective. Nonzero entries in R_L in this section of the trajectory are trying to improve tracking of the $\underline{y}_{FdesB}$ time steps in this region.
- The objective is to get zero tracking error in the world in the high accuracy part of the desired trajectory, in spite of the fact that one's model is imperfect.
- The standard ILC problem is to learn in iterations to converge to zero tracking throughout a trajectory, but people often want to use it to address point-to-point problems instead of tracking problems, to produce accuracy at the endpoint. Doing so asks for high accuracy throughout, when one is only interested in the endpoint. The method developed here makes ILC actually address the endpoint problem as an endpoint problem, extending the range of true applicability of ILC to this important class of systems.

4.6 Numerical Experiments

Consider the same system as in Equation 2.7. Suppose this is fed by a zero order hold running at $100Hz$, and one wants to execute an 8 second trajectory that oscillates at 2 Hz given in continuous time form as $y_d = 1 - cos(4\pi t)$. First we ask for zero error tracking this trajectory every other time step using the cost Equation 4.14 with $R = 0$. The top of 4.1 shows this control penalty for all 800 time steps of control action, the second plot shows the desired trajectory at the addressed time steps, every other step, so the plot goes to 400. The third plot is the tracking error at the addressed time steps which is around $-300dB$ which is a numerical zero. The control action in the bottom plot shows that the control action is not unstable, while the inverse of the discrete time transfer function $G(z)$ equivalent to the $G(s)$ above, has one zero outside the unit circle giving a solution of the inverse system that is a constant times -3.7 to the power of the time step, for the asymptotic location, and around -3.2 for this sample rate.

One is interested to know what the tracking is like for the time steps that are not addressed. The control action being chosen for these times steps does not know what the desired trajectory is, and it is only defined for us because the desired output was given in continuous time so that we could sample at these unaddressed time steps and find the error. This is presented in Figure 4.2 The top plots labeled as actual, is the output of the control system for all 800 time steps, and no anomalous behavior is visible for the unaddressed time steps. The second plot is the desired trajectory sampled every other time step used by the algorithm to produce the control action. The bottom plot gives the "error" at the unaddressed time steps, the algorithm makes no attempt to control the size of this error. It

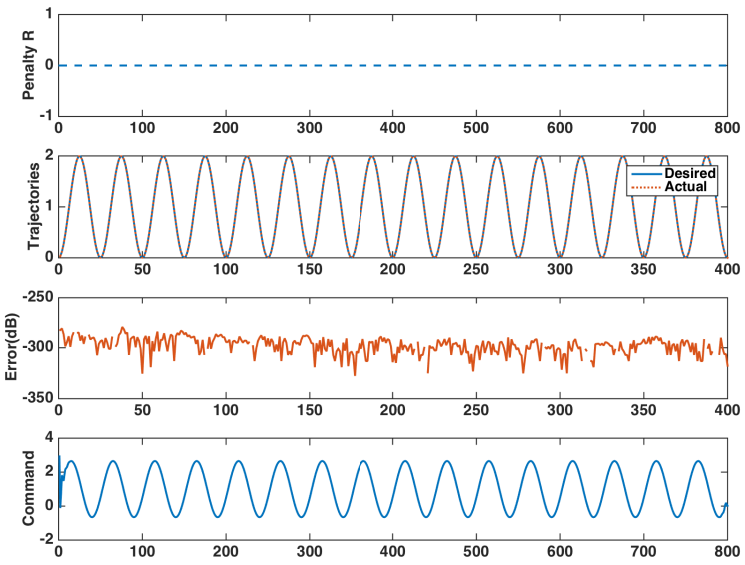

Figure 4.1: Perfect tracking with $R = 0$

is seen to be around $-120dB$ to $-140dB$, corresponding to 10^{-6} or 10^{-7} . At the beginning of the trajectory this "error" is larger but decays reasonably quickly.

Figure 4.3 investigates what happens when the penalty R is changed from one to zero (top plot), suddenly asking for zero tracking error. The output trajectory is shown in the second plot which only tracks as well as the quadratic cost compromise between tracking and control effort dictates for this value of R when it is unity. When the R is set to zero, the error in following the trajectory is large, and the stable inverse control action to graphical accuracy starts giving what looks like zero error reasonably quickly, but the control action has to try to produce something like an impulse function to instantaneously move to zero tracking error (bottom plot). Looking at the tracking error on the 3^{rd} plot, one sees that the error is decreasing exponentially toward the numerical zero around $-300dB$ and needs roughly 50 time steps to reach this zero.

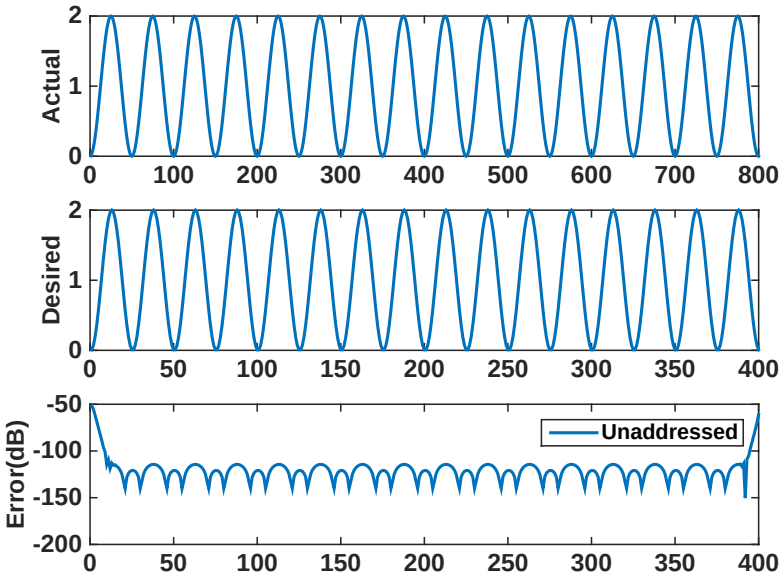

Figure 4.2: Examining the error at unaddressed time steps

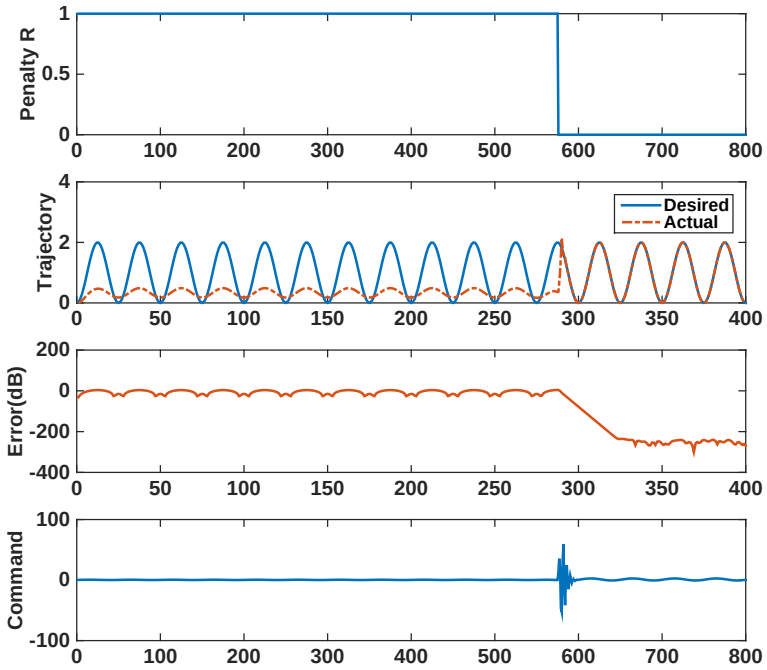

Figure 4.3: Examining the behavior when R changes from 1 to 0 suddenly

Figure 4.4 makes R decrease linearly from 1 to zero over 300 steps, which is seen to improve performance toward zero error in a smooth way, with the wiggles observed previously now being much smaller. This plot investigates using the method of this paper for endpoint control. Very often one wants the output to go to some final position and just stay there afterward. In order to address this desire, the trajectory has been altered, asking the output to be zero at the end at 350 steps, and stay there for 50 more time steps. This demonstrates the effectiveness of use of the algorithm developed here for endpoint problems. The same comments about an impulse function apply to converting to staying at the endpoint. The desired trajectory and its first derivative are zero, but the second derivative is not, and to make a step change in continuous time $u(t)$ would require an impulse, and this produces some wiggles.

In order to reduce the need for such wiggles in the control action, one can make a smoother transition to the section asking for zero tracking error. Figure 4.5 considers having a zero tracking error section in the middle of the trajectory instead of at the end, and uses a 5th order polynomial $10t^3{-}0.15t^4 + 0.0006t^5$ applied to the appropriate time steps for both going onto the zero error portion and coming off of it, with time reversed between the two cases. This is designed to be zero with zero first and second derivatives at the start, and have the final value desired at the end, but with zero first and second derivatives. Clearly this approach is effective at addressing the is- sue of wiggles in the control action when starting onto or getting off of the zero tracking error portion of the trajectory. Figure 4.6 is an example of applying the iterative learning control approach developed here, to learn to get zero error in the zero tracking error portion of the trajectory when the model used by the control algorithm has undamped natural frequency of 5.9Hz as before, but in the real

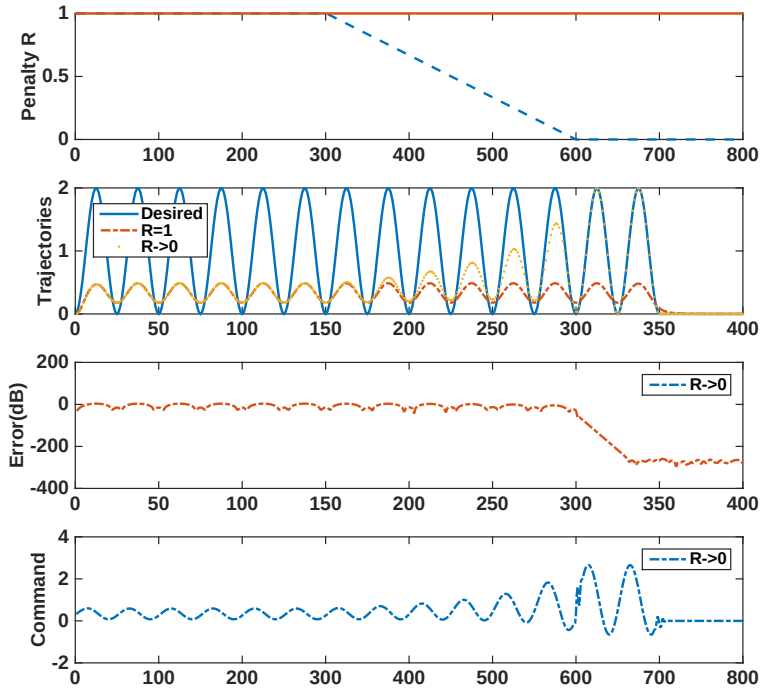

Figure 4.4: Change R linearly from 1 to 0, and addressing the problem of staying at the endpoint

world, this frequency is 8Hz. The new penalty on the change in control from one run to the next set to $R_L = 0.0001$. After 10 iterations for learning, the error in the high accuracy part of the trajectory has reached a numerical zero in the real world model in spite of the algorithm using a stable inverse of an incorrect model.

4.7 Conclusions

This chapter uses a new stable inverse that can produce zero tracking error for discrete time systems having a zero of zeros outside the unit circle, which occurs for nearly all systems above second order after conversion to discrete time. The new stable inverse offers zero

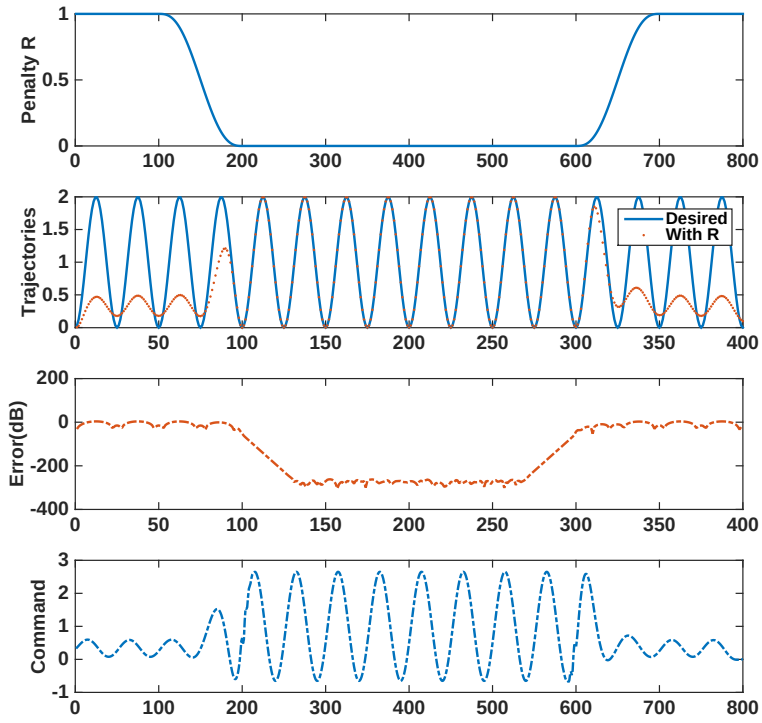

Figure 4.5: Zero tracking in the middle of a trajectory with polynomial conversion of R between 1 and 0

tracking error for every time step in exchange for using multiple zero order holds between steps. Problems are addressed where one wants high precision tracking for a portion of the total time interval. The trajectory in other portions makes use of the effective routine control tracking method that minimizes a quadratic cost function as in linear quadratic regulator tracking. The method creates a unified quadratic cost method to produce both parts of the trajectory. When using the design method, one should build in a transition zone to get onto the zero tracking error part of the trajectory, and also when leaving this part of the trajectory. After designing the desired trajectory, a method of using iterative learning control is presented to learn to achieve zero tracking in the world (limited by the noise floor in hardware), while the initial control action is based on an imperfect model. An important

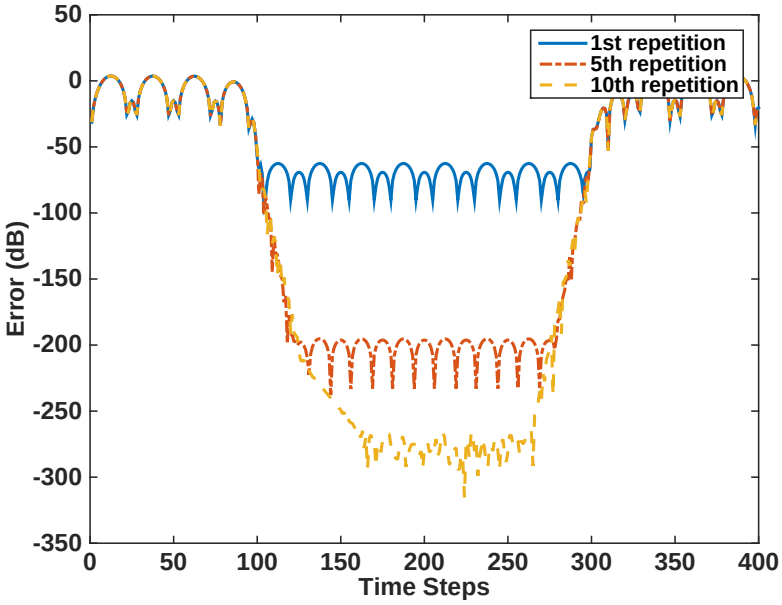

Figure 4.6: ILC learning to achieve zero error in the middle of a trajectory when the stable inverse used an inaccurate model

aspect of the design approach developed here is that it allows iterative learning control to address the problem of learning to reach a desired endpoint, when the formulation of learning control is created to obtain zero tracking throughout a desired trajectory.

Chapter 5

Linear Model Predictive Control Design with Stable Inverse

In the previous chapter, we have introduced a method of producing a stable inverse of systems that have zeros outside the unite circle so the full inverse is unstable. The methods produce zero tracking error expect at one or more initial time steps. The purpose of this chapter is to outline a set of uses for such a stable inverse in control applications, including LMPC, and LMPC applied to Repetitive Control RC-LMPC, and a generalized form of one step ahead control. An important characteristic is that the approach has the property that it can converge to zero tracking in a small number of time steps, which is finite time convergence instead of asymptotic convergence as time tends to infinity. The majority of discrete time systems in the world obtained from discrete differential equation systems fed by zero order hold have unstable inverses. The existence of a stable inverse that produces zero tracking error at addressed time steps opens up a range of new possibilities in control

theory.

5.1 Introduction

Repetitive Control (RC) and Iterative Learning Control (ILC) are control methods that seek to converge to zero error in repeating situations such as periodic command or periodic disturbance. This chapter seeks to make a preliminary investigation of synergistic cross fertilization between several control design methods for single-input, single-output systems - RC, ILC, linear model predictive control (LMPC), and one step ahead control - by making use of a recently developed stable inverse.A common objective in this work is to produce control actions that converge to zero tracking error in a finite number of time steps, instead of converging asymptotically. These different control design methods address different control objectives which are described below. The intention is to describe the range of possible applications of the stable inverses developed to various kinds of control designs and objectives. One Step Ahead Control considers higher order discrete time models with a natural time delay through the system of one time step, for example given the desired output in the next time step, and knowing all previous inputs and outputs, one simply solves for the current input to produce the desired output in the next step. This approach works for systems with a stable inverse, but usually produces far too aggressive control action in practice. Equally important is the fact that most discrete time systems coming from continuous time systems fed by a zero order hold, have an unstable inverse. Here we seek to use a stable inverse theory developed by the authors for ILC, to address the second issue. Then we look for ways to slow down the convergence to meet actuator constraints, but still

converge to zero error in a finite number of steps.

5.2 The Use of the Stable Inverse Approach

This chapter seeks to give a preliminary assessment of the use of the stable inverse approach developed by the authors and co-workers, to address 3 different Objectives.

- This objective asks to create an approach to LMPC addressing tracking problems. It is assumed that the desired trajectory is always feasible. The approach seeks zero tracking error after a small number of time steps, while honoring the actuator constraints, for any chosen trajectory. It is anticipated that one can achieve this even when one makes changes in the future desired trajectory going to the prediction horizon, so that one achieves zero tracking error following an arbitrary trajectory requiring much more limited knowledge of the future desired trajectory. This can make LMPC more versatile allowing it to react more quickly to changes of plan.
- RC aims to reach zero tracking error asymptotically for a periodic trajectory and/or with a periodic disturbance. This Objective seeks to reach zero tracking error in a small number of time steps, honoring actuator limitations assuming that the desired trajectory is feasible. It seeks to do so by using the LMPC formulation together with the one period back data RC normally uses. LMPC makes plans for the finite time interval to the horizon, and therefore one can make use of the finite time stable inverse.
- Develop a way to make a generalized one-step ahead control apply to systems with an unstable inverse. Then suggest ways to slow the convergence to reach zero error in a

number of time steps but satisfy actuator saturation constraints. Many issues need to be addressed, but this allows one to have zero tracking error at addressed time steps for arbitrary trajectories. There is a natural extension of this Objective to indirect adaptive control

Objective 1: LMPC that converges to zero tracking error in a small number of time steps

LMPC is the obvious applications for the stable inverses of the previous section. LMPC always assumes that you know the trajectory that you plan to use from the present to the prediction horizon. As long as the prediction horizon is long enough to create a stable inverse, one could apply the inverse to compute the LMPC control action. This is an advance, since using the system inverse in LMPC was not possible for the preponderance of real world systems, those that have unstable inverses due to sampling zeros.

Without Actuator Constrain: Consider the third order system of 2.7 with pole excess of 3 fed by a zero order hold. Asymptotically as the sample time interval tends to zero the two zeros tend to -3.732 and $-1/3.732$,the first of which makes the inverse model unstable. By either the first or second stable inverse method, one skips the first time step. Then the control at time step k for LMPC is planned for all time steps to the prediction horizon, producing zero error 2 steps ahead and thereafter if applied. LMPC then will apply the first two time steps of the planned control. The process repeats after two time steps. The result is zero error every other time step. An issue to address is, what is the behavior at the unaddressed time steps. One expects that when the desired trajectory is rather smooth at the

chosen sample rate, then the minimum norm solution for the control history will produce reasonable control actions and errors at these intermediate steps. Of course, the error should be compared to what would happen using just one zero order hold input over both steps.

Slowing the Convergence: The stable inverse is like one-step ahead control or dead-beat control, and can ask for very large control actions during the initial steps. We assume that the desired trajectory is feasible, and not following the boundary of the feasible region. But the process of getting to the desired trajectory can easily ask for control actions beyond saturation. One method of addressing this while preserving the convergence to zero tracking error every other time step, is apply standard quadratic programming at each time step: Minimize $e_{DF,r}^T(k)e_{DF,r}(k)$ subject to $|u(k+i)| < u_{max}$ for $i = 0, 1, \cdots, r-1$, where $e_{DF,r}(k) = y_{DF,r}(k) - y_{DF,r}^\star(k)$ and $y_{DF,r}^\star(k)$ is the desired future trajectory. By picking $u_{max}(k)$ each time step one can control the convergence rate. Alternatively, one could ask at step k to minimize $e_{DF,r}^T(k)e_{DF,r}(k) + u_{F,r}^T R(k) u_{F,r}(k)$, and adjust $R(k)$ during the transients, according to a line search to keep the control actions feasible. Once near enough the desired trajectory, $R(k)$ can be set to zero producing zero tracking error.

Comment: The methods just discussed produces zero tracking to any arbitrary trajectory known r time steps in advance, and converges to zero error in a finite number of time steps. Standard LMPC does not always produce good tracking performance, because it is always aiming for the desired trajectory, and when the time step advances it aims again, never reaching the objective. The stable inverse used here ad- dresses this issue. Also, one disadvantage of LMPC is that it asks you to know the command you want to executed r

time steps into the future. It appears that the stable inverse solutions above could partially alleviate this property as well. As long as some reasonable choice for r time steps in the future is made, for r large enough to eliminate the instability signatures from the P matrix, then one can have zero error 2 time steps in the future for the 3rd order example, and repeat this. The issue of course is, will the 2 control actions that gets to zero error at the second step, do something undesirable in the intermediate point. One expects to need a certain level of smoothness. But there seems to be the possibility to have zero error following an arbitrarily changing command with a short preview time.

Objective 2: Model Predictive Repetitive Control That Converges in a Small Number of Iterations

Repetitive control has two properties to try to incorporate into LMPC. The standard RC problem aims to converge to zero tracking error of a periodic command or in the presence of a periodic disturbance, or both when both have the same period. The stable inverse solutions developed here do not include a disturbance unless one knows the disturbance time history and includes it in the inverse equation. On the other hand RC makes a difference between the inputs and outputs one period back and in the present time, and this difference eliminates the disturbance from the dynamics needed. All that must be known is the period of the disturbance. The second objective in RC is to converge to zero error in the real world, not zero error in our model. This objective relies on having enough stability robustness to model error to converge to zero error in spite of an imperfect model. Here we can make RC-LMPC that handles the disturbance, and reaches zero tracking error in a small number

of steps assuming the model is right. It requires that the prediction horizon r be shorter than one period. Stability robustness is an issue to be investigated, but provided the real world is inside the convergence region, RC will learn a modified command to give to the stable inverse such that it produces zero tracking error. This convergence is likely asymptotic. Some existing literature considers the relationship of RC and LMPC and develops a combination, References [13], [40]. The approach here differs in that it offers to converge to zero tracking error in a small number of time steps with a correct model, and makes use only of input-output data without asking for knowledge of the state. To a produce the RC-LMPC law, write the dynamic equations using the LMPC model at the present time step k, and also one period back, where one period is p time steps. Define a difference operator δ^p giving the value of some variable at the present time, minus that variable one period back. Then

$$\begin{aligned} e_{DF,r}(k-p) &= P_{D1}u_{P,s}(k-p) - P_{D2}y_{P,s}(k-p) + W_D u_{F,r}(k-p) - y^{\star}_{DF,r}(k-p) \\ e_{DF,r}(k) &= P_{D1}u_{P,s}(k) - P_{D2}y_{P,s}(k) + W_D u_{F,r}(k) - y^{\star}_{DF,r}(k) \end{aligned} \tag{5.1}$$

Because we seek to track a periodic trajectory $y^{\star}_{DF,r}(k) = y^{\star}_{DF,r}(k-p)$. And if we want to produced zero error fast we ask for $e_{DF,r}(k) = 0$ to eliminate the error observed in the previous period. Then the control update law using the stable inverse is

$$\delta^p u_{F,r}(k) = -W_D^{-1}[e_{DF,r}(k-p) + P_{D1}\delta^p u_{P,s}(k) - P_{D2}\delta^p y_{P,s}(k)] \tag{5.2}$$

As discussed in the LMPC problem above, one has choices of how to slow the convergence rate to remain within actuator limits. One can solve a quadratic programming problem to

minimize $e_{DF,r}^T e_{DF,r}(k)$ subject to actuator inequality constraints each time step. Or one can minimize $e_{DF,r}^T(k) e_{DF,r}(k) + \delta^p u_{F,r}^T(k) R(k) \delta^p u_{F,r}$ each time step. The changes in the control actions $\delta^p u_{F,r}(k)$ will accumulate to produce zero error asymptotically, buy by picking $R(k)$ to satisfy the current control limitation by a line search, one satisfies the actuator saturation during convergence, but can set it zero eventually to produce zero tracking error in a finite number of times steps.

Objective 3: Generalized One Step Ahead Control

The above two objectives are immediately achievable. This third objective is a substantial generalization. The objective of one step ahead control is to have zero error in the next step no matter what one wants to do in the next step. In order to apply the stable inverses above, one needs to define a desired trajectory r steps into the future where r has a lower limit based on how many time steps are needed in P_D for it to not exhibit the signatures of an unstable inverse. But it is clear that one can plan many possible future trajectories which will define a stable control sequence and produce zero error at the second time step, for the 3^{rd} order pole excess example. The issue will be whether the action at the first time step is something reasonable. Guidelines can be produced to say how to extend the trajectory into the future and have the control action for the unaddressed time step be something acceptable. Again, one step ahead control can easily ask for unreasonably large control actions for arbitrary changes in the output in the next step, and zero error every other step can be doing the same. Again we can formulate methods to slow the convergence as was done in both of the previous Objectives, e.g. by making use of a quadratic program updated in real time to

satisfy actuator limits. If these issues are addressed, it opens the door to performing indirect adaptive control replacing the one step ahead algorithm, and making it apply to not just the small number of discrete time systems with stable inverse, but apply in general.

CONCLUSION

Repetitive Control nominally asks for zero error of a control system executing a periodic command, or in the presence of a periodic disturbance. This means zero error at the fundamental frequency of the given period, and all harmonics up to Nyquist frequency. One normally needs to use a frequency cutoff for a number of possible reasons. A cutoff filter can be needed for stability robustness to high frequency model error. Limiting the frequency upper limit in the RC cost function can be use to improve the accuracy of the compensator $F(z)$ in modeling the frequency response of $G^{-1}(z)$ in the region below the learning cutoff frequency. The choice of these two cutoff's should be coordinated. The penalty W_a in the cost function 2.6 is included to directly decrease the gain magnitudes. But it can also act like a frequency cutoff. Improve robustness to error in a chosen frequency range can be achieved by slowing the learning rate, but when model error is too large for learning to succeed one must use the primary cutoff filter $H(z)$. This dissertation seeks to show the choices one needs to make, and how they should be made, to create RC that converges to as small an error as possible.

Learning more slowly can improve the robustness to model error, and the laws devel-

oped in ILC have good robustness properties for this reason. In Chapter 2, we generate RC laws that are analogous to these ILC laws, and the resulting RC laws improve the robustness while have fast learning rate in the low frequency range.

Many real-world situations have a linear system with input through a zero order hold and sampled output. Often one knows the desired output and would like to solve the inverse problem of finding that input that produces this output. For the majority of physical systems this results in an unusable input that grows exponentially with time and alternates sign each time step. Recent results demonstrated a new stable inverse method produced by allowing two or more zero order holds between each time step for which ones at which one asks for zero error. This forms a kind of generalized hold. The existence of a stable inverse that produces zero tracking error at addressed time steps opens up a range of new possibilities in control theory. Chapter 3 addresses problems where one wants high precision tracking for only a portion of the total time interval. The trajectory in other portions makes use of the effective routine control tracking method that minimizes a quadratic cost function as in linear quadratic regulator/tracking. The method creates a unified quadratic cost method to produce both parts of the trajectory. When using the design method, one should build in a transition zone to get onto the zero-tracking error part of the trajectory, and also when leaving this part of the trajectory. After designing the associated desired trajectory, a method of using iterative learning control is presented to learn to achieve zero tracking in the world (limited by the noise floor in hardware), when the initial control action is based on an imperfect model. An important aspect of the design approach developed here is that it allows iterative learning control to address the problem of learning to reach a desired endpoint, when the formulation of learning control is created to obtain zero tracking throughout a

desired trajectory.

By using the same stable inverse, Chapter 4 outlines a set of uses for such methods in control applications, including Linear Model Predictive Control (LMPC), and LMPC applied to Repetitive Control RC-LMPC, and a generalized form of one step ahead control. An important characteristic is that the approach has the property that it can converge to zero tracking error in a small number of time steps, instead of converging asymptotically as time tends to infinity.

There are a number of topics for future research. One topic to generalize the guide for RC design to multi-input, multi-output (MIMO) systems. The cost function used in SISO can be generalized to MIMO systems, and the resulting compensator will have similar concerns as discussed in Chapter 2. To robustify the design to unmodeled high-frequency dynamics, a MIMO zero-phase low pass FIR filter will be necessary. Another future topic is to apply the design methods based on stable inverse to nonlinear problem. The first approach is linearization. But when the system is linearized about a trajectory, the system equations become linear with time varying coefficients. So we need to develop methods to handle systems with time varying or periodic coefficients.

A rigorous proof for the RC stability condition have been given in previously, it uses Nyquist stability like thinking. But for nonlinear systems, perhaps the stability condtion ca be generalized to nonlinear system by employing the logic behind the Popov criterion or equivalent circle criterion.

References

[1] Inoue, T., Nakano, M., and Iwai, S., 1981. "High accuracy control of a proton synchrotron magnet power supply". *Proceedings of the 8the World Congress of IFAC*, pp. 216–221.

[2] OMATA, T., NAKANO, M., and INOUE, T., 1984. "Application of repetitive control method to multivariable systems". *Transactions of the Society of Instrument and Control Engineers,* ***20***(9), pp. 795–800.

[3] Middleton, R. H., Goodwin, G., and Longman, R., 1989. "A method for improving the dynamic accuracy of a robot performing a repetitive task". *International Journal of Robotics Research,* ***8***, pp. 67–74.

[4] Hara, S., Omata, T., and Nakano, M., 1985. "Synthesis of repetitive control systems and its application". In *1985 24th IEEE Conference on Decision and Control* (Dec), pp. 1387–1392.

[5] Nakano, M.and Hara, S., 1986. *Microprocessor-Based Repetitive Control*. Springer Netherlands, pp. 279–296.

[6] Tomizuka, M., Tsao, T.-C., and Chew, K.-K., 1989. "Analysis and synthesis of discrete time repetitive controllers". *Journal of Dynamic Systems, Measurement, and Control,* ***111***, pp. 353–358.

[7] Longman, R. W., 2000. "Iterative learning control and repetitive control for engineering practice". *International Journal of Control,Special Issue on Iterative Learning Control,* ***73***, pp. 930–954.

[8] Longman, R. W., 2010. "On the theory and design of linear repetitive control systems". *European Journal of Control, Special Section on Iterative Learning Control,* ***16***, pp. 447–496.

[9] Panomruttanarug, B., and Longman, R. W., 2004. "Repetitive controller design using optimization in the frequency domain". In *Proceedings of the 2004 AIAA/AAS Astrodynamics Specialist Conference* (Providence, RI, August), AIAA/AAS Astrodynamics Specialist Conference.

[10] J. Bao, R. L., 2010. "Unification and robustification of iterative learning control laws". *Advances in the Astronautical Sciences,* ***136***, pp. 727–745.

[11] Li, T., and Longman, R. W., 2015. "Robustification of iterative learning control produced by multiple zero order holds and initial skipped steps". *Advances in the Astronautical Sciences,* ***155***, pp. 1311–1330.

[12] Shi, Y., and Longman, R. W., 2012. "The influence on stability robustness of compromising on the zero tracking error requirement in repetitive control". *The Journal of the Astronautical Sciences,* ***59***(1), Jun, pp. 441–458.

[13] K. Chen, R. W. L., and Phan, M., 2006. "On the relationship between repetitive control and model predictive control". *Proceedings of the 2006 AIAA/AAS Astrodynamics Specialist Conference*.

[14] LeVoci, P. A., and Longman, R., 2004. "Frequency domain prediction of final error due to quantization in learning and repetitive control". *Advances in the Astronautical Sciences,* ***116***, pp. 1311–1330.

[15] B. Panomruttanarug, R. L., and Phan, M., 2012. "Designing stable iterative learning control systems from frequency based repetitive control designs". *Advances in the Astronautical Sciences,* ***142***, pp. 2893–2912.

[16] Panomruttanarug, B., and Longman, R. W., 2006. "Frequency based optimal design of fir zero-phase filters and compensators for robust repetitive control". *Advances in the Astronautical Sciences,* ***123***, pp. 219–238.

[17] Bao, J., and Longman, R., 2008. "Enhancments of repetitive control using specialized fir zero-phase filter designs". *Advances in the Astronautical Sciences,* ***129***, pp. 1413–1432.

[18] Y. Shi, R. L., and Phan, M., 2010. "An algorithm for robustification of repetitive control to parameter uncertainties". *Advances in the Astronautical Sciences,* ***136***, pp. 1953–1966.

[19] Phan, M. Q., Longman, R. W., Panomruttanarug, B., and Lee, S. C., 2013. "Robustification of iterative learning control and repetitive control by averaging". *International Journal of Control,* ***86***(5), pp. 855–868.

[20] Arimoto, S., Kawamura, S., and Miyazaki, F., 1984. "Bettering operation of robots by learning". *Journal of Robotic Systems,* ***1***(2), pp. 123–140.

[21] Jang, H. S., and Longman, R. W., 1996. "Design of digital learning controllers using a partial isometry". *Advances in Astronaut Science,* ***93***, pp. 137–152.

[22] Phan, M. Q., and Frueh, J. A., 1998. *System Identification and Learning Control*. Springer US, Boston, MA, pp. 285–310.

[23] Owens, D. H., 2016. *Norm Optimal Iterative Learning Control*. Springer London, London, pp. 233–276.

[24] Chen, K., and Longman, R. W., 2003. "Creating short time equivalents of frequency cutoff for robustness in learning control". *Advances in Astronaut Science,* ***114***, pp. 95–114.

[25] Avrachenkov, K. E., and Longman, R. W., 2003. "Iterative learning control for over-determined, under-determined, and ill-conditioned systems". *International Journal of Applied Mathematics and Computer Science,* ***13***, pp. 113–122.

[26] Moore, K. L., and Xu, J.-X., 2000. "Editorial: Special issue on iterative learning control". *International Journal of Control,* ***73***(10), pp. 819–823.

[27] Åström, P., Hagander, P., and Strenby, J., 1980. "Zeros of sampled systems". *Proceedings of the Nineteenth IEEE Conference on Decision and Control*, pp. 1077–1081.

[28] LeVoci, P., and Longman, R., 2004. *Intersample Error in Discrete Time Learning and Repetitive Control*. American Institute of Aeronautics and Astronautics, 2017/10/28.

[29] Li, Y., and Longman, R., 2010. "Using underspecification to eliminate the usual instability of digital system inverse models". *Advances in the Astronautical Sciences,* ***135***, 01, pp. 127–148.

[30] Li, Y., and Longman, R., 2010. "Characterizing and addressing the instability of the control action in iterative learning control". *Advances in the Astronautical Sciences,* ***136***, 01, pp. 1967–1985.

[31] Li, T., and Longman, R. W., 2016. *Designing Iterative Learning Control of Non-Minimum Phase Systems to Converge to Zero Tracking Error*. American Institute of Aeronautics and Astronautics, 2017/10/28.

[32] Longman, R. W., and Li, T., 2017. "A new approach to producing a stable inverse of discrete time systems". Proceedings of the 18th Yale Workshop on Adaptive and Learning Systems.

[33] Ji, X., and Longman, R. W., 2017. "Proof of a new stable inverse of discrete time systems". *AIAA/AAS Astrodynamics Specialist Conference*, August.

[34] Devasia, S., Chen, D., and Paden, B., 1996. "Nonlinear inversion-based output tracking". *IEEE Transactions on Automatic Control,* ***41***(7), Jul, pp. 930–942.

[35] Sogo, T., 2002. "Stable inversion for nonminimum phase sampled-data systems and its relation with the continuous-time counterpart". In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* (Dec), Vol. 4, pp. 3730–3735 vol.4.

[36] Bottou, E., and Vapnik, V., 1992. "Local learning algorithms". *Neural Computation,* ***4***, pp. 888–900.

[37] S. J. Oh, R. L., and Phan, M. Q., 1997. "Use of decoupling basis functions in learning control for local learning and improved transients". *Advances in the Astronautical Sciences,* ***96***, pp. 651–670.

[38] Wang, H., Zou, Q., and Xu, H., 2012. "Inversion-based optimal output tracking transition switching with preview for nonminimum-phase linear systems". *Automatica,* ***48***(7), pp. 1364 – 1371.

[39] Panomruttanarug, B., Longman, R., and Phan, M., 2009. "Multiple model robustification of iterative learning and repetitive control laws including design from frequency response data". *Advances in the Astronautical Sciences,* ***134***, 01, pp. 2259–2278.

[40] Cruz, D. M., Normey-Rico, J. E., and Costa-Castelló, R., 2014. "Repetitive model based predictive controller to reject periodic disturbances.". *IFAC Proceedings Volumes,* ***47***(3), pp. 11494 – 11499. 19th IFAC World Congress.